LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

GRAAFISEN SUUNNITELUTYÖKALUN TOTEUTUS

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2013
Antti Terho

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

TERHO, ANTTI: Graafisen suunnittelutyökalun toteutus

Ohjelmistotekniikan opinnäytetyö, 50 sivua

Kevät 2013

TIIVISTELMÄ

Opinnäytetyö tehtiin digitaalisen median tuotantoyhtiö Imager Oy:lle. Opinnäytetyön tarkoituksena oli toteuttaa asiakkaalle web-sovellus peltiprofiilien suunnittelemiseen ja mitoittamiseen. Työkalun tarkoituksena oli siirtyä käsin tehdyistä suunnitelmista ja mitoituksista, sovelluksella piirrettyihin yhteneväisiin ja selkeämmin luettaviin suunnitelmiin.

Web-sovellus toteutettiin käyttäen HTML-, CSS- ja JavaScript-tekniikoita. Piirtoalustana sovelluksessa toimii HTML5 canvas-elementti. Canvas-elementille piirtäminen toteutettiin käyttäen Paper.js JavaScript-kirjastoa. Myös jQuery- ja TweenLite-kirjastot ovat olleet merkittävästi mukana toteutuksessa.

Opinnäytetyön teoriaosuus keskittyy sovelluksessa käytettyjen tekniikoiden ja elementtien esittelyyn. Esiteltyjä tekniikoita ovat HTML, CSS, JavaScript, HTML5, Canvas, CSS3, jQuery, AJAX, JSON, Paper.js sekä TweenLite.

Käytännön osuudessa käydään läpi graafisen suunnittelutyökalun ominaisuuksia ja toteutusta. Suunnitelun kaksi osuutta, piirto ja mitoitus, käydään yksityiskohtaisesti läpi ja pureudutaan tarkemmin niissä tehtyihin ratkaisuihin.

Lopputuloksena sovellukseen saatiin sisällytettyä kaikki halutut ominaisuudet ja sovellusta tullaan tulevaisuudessa testaamaan lisää ja kehittämään asiakkaan toiveiden mukaan.

Asiasanat: Canvas, JavaScript, jQuery, Paper.js, TweenLite

Lahti University of Applied Sciences
Degree Programme in Information Technology

TERHO, ANTTI: Implementation of a graphical designing tool

Bachelor's Thesis in software engineering, 50 pages

Spring 2013

ABSTRACT

This Bachelor's Thesis was made for a digital media production company called Imager Ltd. The purpose of the thesis was to make a web application for designing and dimensioning sheet metal profiles. The objective of the application was to move from hand-drawn designs and dimensioning to consistent and clearly readable plans.

The web application was made by using HTML, CSS and JavaScript techniques. The HTML5 canvas element functions as the drawing platform of the application. Actual drawing was implemented using JavaScript library called Paper.js. The jQuery and TweenLite libraries had a big role in the implementation.

The theoretical part focuses on the introduction of the techniques and elements used. The techniques are HTML, CSS, JavaScript, HTML5, Canvas, CSS3, jQuery, AJAX, JSON, Paper.js and TweenLite.

The practical part contains the introduction of the features and implementation of the graphical design tool. The two sections of the application, design and drawing, are analyzed thoroughly and a closer look is taken at the solutions that were made during the developing process.

As a result, the application included all of the desired properties, and the application will be tested more in the future and developed according to the customer's wishes.

Key words: Canvas, JavaScript, jQuery, Paper.js, TweenLite

SISÄLLYS

1 JOHDANTO

Web-sovellusten määrä HTML5-kuvauskielen myötä on kasvanut viime vuosina huomattavasti ja yhä monimutkaisempia ohjelmia on mahdollista toteuttaa web-sovelluksena. Web-sovelluksella tarkoitetaan ohjelmaa, jolla on web-käyttöliittymä, ja siihen päästään käsiksi mistä vain, kunhan saatavilla on internet-yhteys ja moderni internetselain.

Tämä opinnäytetyö on tehty Imager Oy:lle. Imager Oy on digitaalisen median tuotantoyhtiö, ja se sijaitsee Lahdessa Rautatienkatu 13C:ssä. Imager Oy on perustettu vuonna 2000, ja se on työntekijöidensä yhdessä omistama. Vakituisia työntekijöitä yrityksessä on yhteensä viisi. Karkeasti jaoteltuna yksi työntekijöistä toimii pääsääntöisesti ohjelmointia vaativissa työtehtävissä ja neljä muuta työntekijää suorittavat digitaalisen median tuottamisen printtimediasta vaativaan 3d-mallintamiseen ja audiovisuaaliseen tuotantoon. (Imager.fi 2013.)

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa graafinen suunnittelutyökalu Imager Oy:lle, ja hiottu lopputulos esiteltäisiin myöhemmin loppuasiakkaalle. Suunnittelutyökalun tuli olla monipuolinen, mutta samalla helppokäyttöinen ja yksinkertainen apuväline peltiprofiilien suunnitteluun. Sovelluksen tuli myös pystyä tuottamaan tarvittavat datat profiilin varsinaista valmistusprosessia varten.

Opinnäytetyön teoriaosuudessa tutustutaan yleisesti erilaisiin tekniikoihin, jotka mahdollistivat graafisen suunnitelutyökalun toteutuksen web-sovelluksena. Teoriaosuudessa otetaan myös tarkempi katsaus web-sovelluksessa käytettyihin JavaScript-kirjastoihin. Käytännönosuudessa paneudutaan tarkemmin graafisen suunnittelutyökalun toimintaan ja ominaisuuksiin.

2 SOVELLUKSESSA KÄYTETYT TEKNIIKAT

Graafinen suunnittelutyökalu on toteutettu pääosin käyttäen kolmea web-maailman tekniikkaa: HTML:ää, CSS:ää ja JavaScript:iä. Tässä osuudessa kerrotaan teoriaa siitä, miten kukin tekniikka on mahdollistanut sovelluksen toteutuksen. Kaikkia kolmea päätekniikkaa ei ole tarkoitus käydä läpi kokonaan, vaan kertoa taustatietoa kyseisestä tekniikasta ja selvittää tarkemmin tekniikoiden osia, joita on käytetty graafisen suunnitelutyökalun toteutuksessa.

2.1 HTML

HTML on avoimesti standardoitu hypertekstin merkintäkieli, ja se on lyhenne englanninkielisistä sanoista Hypertext Markup Language. HTML:llä voidaan kuvata hyperlinkkejä sisältävää tekstiä sekä tekstin rakennetta, eli esimerkisi, mikä osa tekstistä on kappaletta ja mikä otsikkoa. HTML:ää on yleisimmin käytetty web-sivujen rakentamiseen. (Wikipedia 2013d.)

HTML-dokumentti koostuu sisäkkäisistä ja perättäisistä elementeistä. Elementillä tarkoitetaan kulmasulkein merkittyä tunnistetta (tag), joka nimeää elementin tyypin. Internetselaimet eivät näytä tunnisteita sellaisenaan vaan käsittelevät niitä ohjeina, joiden mukaan sivun sisältö tullaan jäsentelemään. HTML:n tunnisteet saattavat sisältää attribuutteja, jotka määrittävät elementille lisäominaisuuksia.

```
1  <tunniste mahdollinen_attribuutti="arvo"></tunniste>
2  <div id="iidee"></div>
3  <br />
```

KUVIO 1. HTML-elementti-esimerkki

Kuviossa 1 nähdään esimerkki HTML-elementistä ja sen rakenteesta. Lopputuunnistetta elementissä kuvataan ”/”-merkillä. Eräät HTML-elementit eivät sisällä mitään sisältöä, joten niissä ei tarvita lopetustunnistetta, kuten esimerkiksi rivillä 3 olevassa rivinvaihtoa kuvaavassa tunnisteessa, vaan ”/”-merkki voidaan lisätä aloitustunnisteen loppuun. (Wikipedia 2013d.)

HTML Document Object Model, eli HTML DOM on ohjelmointirajapinta, joka mahdollistaa HTML-tiedostojen sisällön rakenteen ja tyylien dynaamisen muokkaamisen. Kuviossa 2 nähdään esimerkki HTML-dokumentista. (Wikibooks 2013b.)

```
1  <html>
2    <head>
3      <title>HTML DOM Esimerkki</title>
4    </head>
5    <body>
6      <h1>Otsikko</h1>
7      <p>Tämä on kappale!</p>
8    </body>
9  </html>
```

KUVIO 2. HTML-dokumentti-esimerkki

Koko HTML-dokumentti on solmu (node), ja jokainen HTML-elementti on solmu. Elementtien sisällä olevat sisällöt, elementtien attribuutit ja kommentit ovat solmuja. HTML DOM jäsentää HTML-tiedoston puurakenteena. Kuvion 2 esimerkissä html-elementti on dokumentin pää- eli juurisolmu (root). Jokaisella solmulla paitsi juurella on aina yksi isäntäsolmu (parent node), mutta lapsia (children) voi olla rajaton määrä. Html-elementillä on siis kaksi lapsisolmua, elementit head ja body. Head ja body ovat keskenään sisaruksia (sibling). Head-solmulla on lapsisolmu title ja bodyllä solmut h1 ja p, jotka ovat sisaruksia keskenään, koska niillä on yhteinen vanhempisolmu. Title-, h1- ja p-elementeillä on sisällään tekstisolmut. Solmua, jolla ei ole lapsisolmuja, kutsutaan lehdeksi (leaf). (Wikibooks 2013b.)

HTML-puussa jokainen solmu on objekti ja jokaisella objektilla on ominaisuuksia (properties) ja metodeja (method). Näytä objektin metodeja ja ominaisuuksia voidaan käsitellä JavaScriptillä tai jollain muulla ohjelmointikielellä. Ominaisuudet ovat esimerkiksi solmun nimi tai arvo. Metodit ovat tekemistä esimerkiksi delete(), eli poista. (Wikibooks 2013b.)

2.2 CSS

CSS, eli Cascading Style Sheets on pääasiassa (X)HTML-dokumenteille tehty tyylikieli. CSS:ssä dokumentille voidaan määritellä tyyliohjeita, joista yhdessä kootaan esityssäännöt. CSS-dokumentissa annetut säännöt ehdottavat sen, miten esimerkiksi HTML-dokumentin sisältö voidaan esittää. CSS:ssä määritetyt säännöt eivät siis ole ehdottomia. Jotkin määritetyt säännöt voivat kumoutua toisten tyyliohjeiden vastaavilla säännöillä. CSS:n ominaisuuksiin kuuluu muun muassa elementin leveyden ja korkeuden määrittäminen, reunaviivat ja täytteet, listojen muotoilu, fonttien muotoilu, asemointi, ylivuodon käsittely, taulukoiden muokkaaminen ja värien sekä taustojen muokkaaminen. (Wikipedia 2013c.)

CSS määrittelyn perussyntaksi muodostuu valitsimesta, ominaisuudesta ja arvosta. Valitsimella voidaan valita elementti tai elementit, joiden ominaisuuksiin asetetut arvot vaikuttavat. Valitsin voidaan kohdistaa esimerkiksi johonkin tiettyyn elementtiin tai attribuuttiin. Yleisesti käytössä olevat attribuutit, joihin CSS-määrittelyissä viitataan, ovat class ja id. Class-attribuuttiin viitataan CSS-dokumentissa pisteellä ”.” ja ID-attribuuttiin ”#”-merkillä. Valitsimen jälkeen tulevat kaarisulkeet, joiden sisään on mahdollista määritellä useita ominaisuus-arvo-pareja. Ominaisuus ja arvo erotellaan toisistaan kaksoispisteellä ja parit erotellaan toisistaan puolipisteellä. Kuviossa 3 nähdään esimerkki CSS:n syntaksista. (Wikipedia 2013c.)

```
1  valitsin {
2      ominaisuus: arvo1;
3      ominaisuus2: arvo2;
4      ...
5      ominaisuusn:arvon;
6  }
7
8  #iidee {
9      width:100px;
10     height:50px;
11 }
```

KUVIO 3. CSS-syntaksi

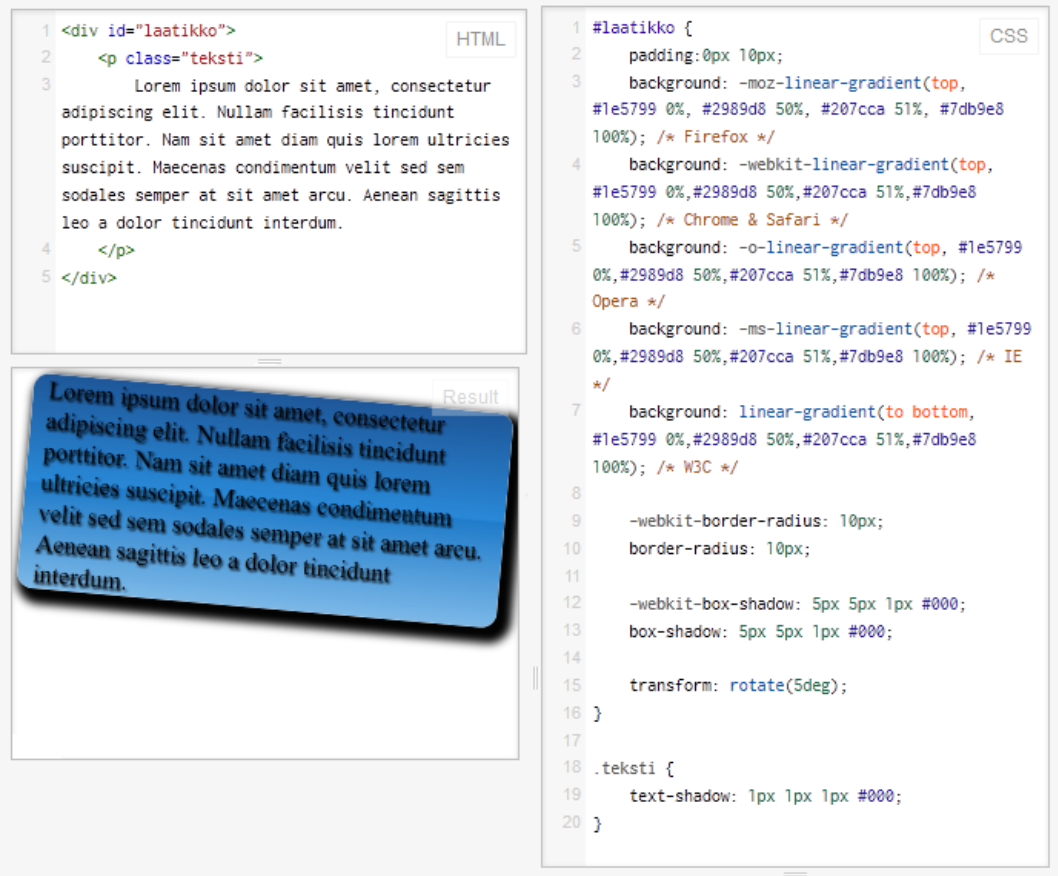
CSS-määritykset lisätään web-sivustolle, joko head-elementin sisällä tehtävän style-elementin sisään tai CSS-määritykset voidaan myös linkittää ulkoisesta dokumentista käyttämällä link-tunnistetta. Kuviossa 4 nähdään rivillä 3 esimerkki ulkoisen CSS-dokumentin linkityksestä ja riviltä 4 alkaen nähdään esimerkki style-tunnisteen käytöstä.

```
1  <html>
2      <head>
3          <link rel="stylesheet" type="text/css" href="polku/css/tiedostoon.css">
4          <style>
5              #id {...}
6              .class {...}
7              elementti {...}
8          </style>
9      </head>
10     <body>
11     </body>
12 </html>
```

KUVIO 4. CSS:n linkitys-esimerkki

CSS3 on CSS:n uusin versio. Versioon kolme on lisätty paljon uusia ominaisuuksia, ja osan ominaisuuksien tarkoituksena on korvata Adobe Flashin ja Java Aplettien tarve nykyisissä web-sovelluksissa. CSS3:n tuki on ollut vielä uusimmissakin internetselaimissa melko heikkoa, ja jotkin uudet ominaisuudet joudutaan asettaamaan eri internetselaimille eri tavoin. (Wikibooks 2013a.)

CSS3:n uusia ominaisuuksia versioon kaksi nähden ovat muun muassa elementtien animointi, kääntäminen, laatikoiden muuntaminen, fonttien venytykset, tekstin kelaaminen, rivien määrä ja taustat sekä puheominaisuudet. CSS3 mahdollistaa myös laatikoiden reunojen pyöristämisen border-radius-arvoa muuttamalla, joka jouduttiin ennen tekemään kuvia hyväksikäyttäen. Elementeille ja tekstille on CSS3:n myötä mahdollista asettaa erilaisia varjoja muuttaen box-shadow- ja text-shadow-arvoja. Tämä myös vähentää omalla tavallaan kuvien käyttöä nyky web-sovelluksissa. Gradientit eli liukuvärit ovat myös CCS3:n käytännöllisimpiä uudistuksia. (Wikibooks 2013a.)



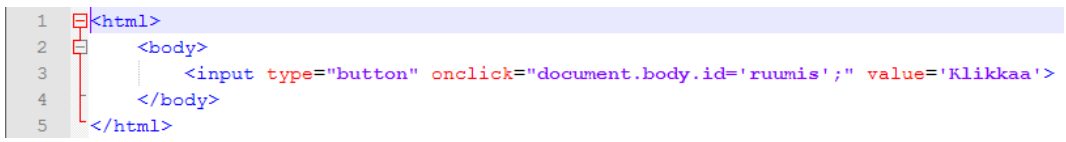
KUVIO 5. CSS3-esimerkki

Kuviossa 5 nähdään esimerkki CSS3-määrittelyistä ja sen uusista ominaisuuksista. Vasemmalla ylhäällä on HTML-osuus, jota muokataan oikealla olevalla CSS3-määrittelyillä. Oikealla CSS-osuudessa valitaan valitsimella aluksi id ”#laatikko”, jolle näön vuoksi laitetaan rivillä kaksi täytettä (padding), minkä jälkeen rivillä kolme, neljä, viisi, kuusi ja seitsemän asetetaan ensimmäinen CSS3:n uusi arvo taustakuvaksi, eli linear-gradient (liukuväri). Aluksi määritellään, mistä gradient alkaa ja sitten värit prosentein. Tässä myös nähdään esimerkki siitä, miten gradient-ominaisuus joudutaan määrittämään eri internetselaimilla eri tavoin. Firefox käyttää etuliitettä –moz, chrome ja safari –webkit, Opera –o, joka muuttunee lähiaikoina –webkit:iin, ja Internet Explorer –ms sekä viimeisenä W3C:n asettama standardiarvo ilman etuliitettä. Rivillä yhdeksän ja kymmenen asetetaan kahdella eri tapaa arvo border-radius, mikä pyöristää #laatikon kulmia 10 pikselin verran. Rivillä 12 ja 13 box-shadow attribuutti asettaa mustan varjon laatikolle ja rivillä 15 oleva transform: rotate(5deg) kääntää laatikkoa viisi astetta vasemmalle. Tämän jälkeen valitaan luokka ”.teksti”, minkä ominaisuuden text-shadow arvoksi määritellään 1px 1px 1px #000, eli musta varjo tekstille. Result-otsikolla olevasta laatikosta nähdään HTML- ja CSS-koodien tuottaman tuloksen.

2.3 JavaScript

JavaScript on Netscapen kehittämä oliopohjainen komentosarjakieli, jolla saadaan lisättyä erilaisia dynaamisia toimintoja web-sivustoille. JavaScript pohjautuu EcmaScript-standardiin. JavaScriptin suorittaminen tapahtuu suoraan käyttäjän omassa internetselaimessa, mikä tarkoittaa sitä, että se on mahdollista kytkeä pois päältä, ja tämä on otettava huomioon kehittäessä web-sovelluksia. Javascriptiä ei tule sekoitta Javaan, vaan JavaScriptin syntaksi muistuttaa pikemminkin C-ohjelmointikieltä. JavaScriptin ohjelmoinnin helpottamiseen on julkaistu useita JavaScript-kirjastoja. (Wikipedia 2013f.)

JavaScriptillä voidaan siis dynaamisesti muokata web-sivuston sisältöä ja lisätä toiminnallisuutta siihen. Aiemmin esitellyssä HTML DOM:ssa solmu(node) olioita voidaan käsitellä JavaScriptillä. DOM:issa voidaan liikkua JavaScriptin avulla ja solmujen ominaisuuksia on mahdollista hakea ja muuttaa sen avulla. Solmu voidaan hakea dokumentista document.getElementByID(”ID tähän”)-metodilla ja elementin sisältöä voidaan muokata esimerkiksi innerHTML=”uusi sisältö”; ominaisuuden avulla. HTML-elementeille on myös mahdollistä lisätä tapahtumankäsittelijöitä(event handler), joiden avulla voidaan suorittaa JavaScript-toimintoja tapahtuman tapahtuessa. Tälläisiä tapahtuman käsittelijöitä ovat esimerkiksi onclick, load, change ja mouseover. Onclick-tapahtumalle määritetty toiminto suoritetaan, kun elementtiä klikataan. Load-tapahtuman toiminto suoritetaan, kun kyseinen elementti on ladattu. Change-tapahtuman toiminto suoritetaan, kun esimerkiksi HTML input-elementin arvo vaihtuu. Mouseover-tapahtuman toiminto suoritetaan, kun hiiren kursori viedään elementin päälle. Tapahtumankäsittelijä asetetaan elementille kuviossa 6 nähtävän esimerkin tapaan. (Wikibooks 2013b.)



KUVIO 6. Tapahtumankäsittelijä

Kuvion 6 esimerkissä rivillä 4 input-elementille asetetaan onclick-tapahtumankäsittelijä, joka elementtiä painettaessa muuttaa dokumentin body elementin id attribuutin arvoksi ”ruumis”.

JavaScriptiä voidaan siis kirjoittaa periaatteessa kaikkien HTML-elementtien yhteyteen, mutta suurin osa JavaScriptistä kirjoitetaan joko script-elementin sisään tai erilliseen tiedostoon, joka linkitetään web-sivustolle head-elementissä. Erillisen tiedoston linkitys tapahtuu käyttäen script-elementtiä, jonka attribuutin src arvoksi asetetaan polku kyseiseen tiedostoon ja attribuutin type arvoksi asetetaan ”text/javascript”. (Wikipedia 2013f.)

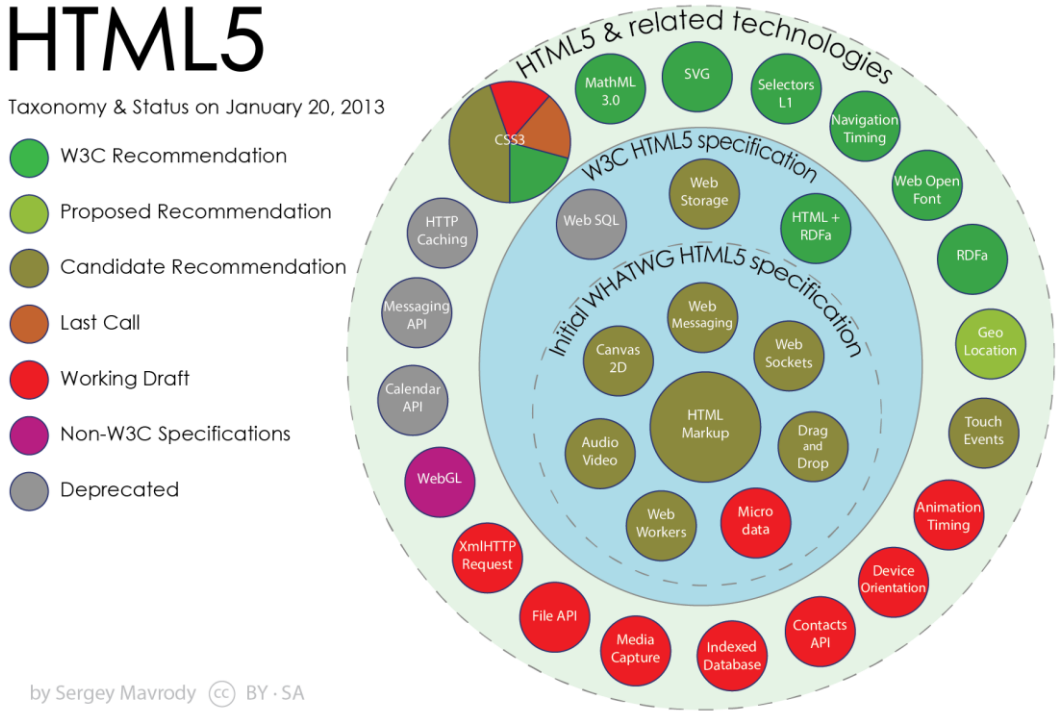
2.4 HTML5

HTML5-kieli, eli Hyper Text Markup Languagen versio 5, on nykypäivän tarpeita tyydyttävä uusi päivitetty versio verkkosivujen tekemiseen yleisesti käytetystä vuonna 1999 julkaistusta HTML 4.01 -kuvauskielestä. HTML5 on kuitenkin vielä keskeneräinen toteutus. Yleisimpien internetselainten, kuten Internet Explorer, Mozilla Firefox, Google Chrome, Opera ja Safari, uusimmat versiot tukevat monia uusia HTML5:n elementtejä ja API:a. HTML5:n kehitys tapahtuu yhteistyössä World Wide Web Consortiumin ja Web Hypertext Application Technology Working Groupin kanssa. (W3Schools 2013b.)

Kiinnostavimpia uusia ominaisuuksia HTML5-kuvauskielessä ovat esimerkiksi <canvas> elementti, joka mahdollistaa 2D- ja nykytekniikoilla myös 3D-grafiikan piirtämisen verkkosivulle. Lisäksi <video> sekä <audio> elementit liikkuvan kuvan ja äänen toistamiseen. HTML5 tukee myös paikallista tallennusta (local storage), mikä mahdollistaa pienien datamäärien tallentamisen suoraan käyttäjän selaimeen. HTML5:stä löytyy myös uudet sisältöä kuvaavat elementit, kuten <header>, <footer>, <article>, <nav> ja <section>. (W3Schools 2013b.)

HTML5-sanalla viitataan nykyään yleisesti moderneihin web-tekniikoihin, joihin kuuluvat HTML5:n uudistusten lisäksi CSS3:n uudistukset sekä erinäiset ohjelmointirajapinnat, eli API:t. HTML5-sovelluskehityksessä ohjelmointikielenä toimii yleensä JavaScript ja muotoilussa käytetään CSS3:a, eli itse HTML5:n

osuus saattaa jäädä usein melko pieneksi, eikä versio viiden uudistuksia ole välttämättä tarve käyttää juuri lainkaan. (Wikipedia 2013d.)



KUVIO 7. Mitä kaikkea HTML5 on Sergey Mavrodyn mukaan 2013 (Wikipedia 2013e)

Kuviossa 7 on esitetty Sergey Mavrodyn näkemys siitä, mitä HTML5 on ja mitä se pitää sisällään vuonna 2013. Kuviossa 7 Vihreällä värillä merkityt ympyrät kuvastavat W3C ehdotuksia. Vaaleanvihreällä merkityt ympyrät kuvastavat ehdotettuja ominaisuuksia. Oliivinvihreällä merkityt ympyrät kuvastavat suositusehdokkaita. Punaisella merkityt ympyrät kuvastavat luonnoksia. Lilat ympyrät kuvastavat W3C:ssä ei-spesifioituja ominaisuuksia ja harmaalla merkityt ympyrät ovat vanhentuneiden ominaisuuksien merkkejä. Kuvion 7 sisimmäisessä ympyrässä nähdään heti HTML Markupin ulkopuolella Canvas 2D, joka toimii keskeisessä osassa myös graafisessa suunnittelutyökalussa.

2.5 HTML Canvas

Canvas on HTML5-elementti, joka mahdollistaa grafiikan piirtämisen dynaamisesti ohjelmoimalla. Ohjelmoiminen tapahtuu yleensä käyttämällä

JavaScript-kieltä. Canvas-elementtiä voidaan käyttää esimerkiksi peleissä, diagrammeissa, animaatioissa sekä monissa muissa 2D-grafiikkaa vaativissa sovelluksissa. (Wikipedia 2013b.)

Canvas-elementti on siis vain säiliö, joka näyttää siihen JavaScriptillä piirretyn sisällön. JavaScriptissä oleva getContext()-metodi palauttaa objektin, joka tarjoaa tarvittavat metodit ja ominaisuudet canvas-elementille piirtämistä varten. Metodeja, joita getContext()-metodin palauttama objekti tarjoaa, ovat muun muassa seuraavat: createLinearGradient(), createPattern(), createRadialGradient(), rect(), fillRect(), strokeRect(), clearRect(), fill(), stroke(), beginPath(), moveTo(), closePath(), lineTo(), arc(), arcTo(), scale(), rotate(), translate(), transform(), fillText(), strokeText(), drawImage(), toDataURL(), save() ja restore(). (W3Schools 2013a.)

createLinearGradient()-metodi luo lineaarisen liukuväriobjektin, jota voidaan käyttää värin tapaan täyttämään esimerkiksi ympyröitä tai suorakulmioita. createPattern()-metodi toistaa haluttua elementtiä, esimerkiksi kuvaa haluttuihin suuntiin ja luo siitä kuvion, jota voidaan käyttää gradienttien tapaan täyttämää esimerkiksi ympyröitä. createRadialGradient()-metodi luo pyöreän liukuväri-objektin, jota myös voidaan käyttää objektien täytteenä. (W3Schools 2013a.)

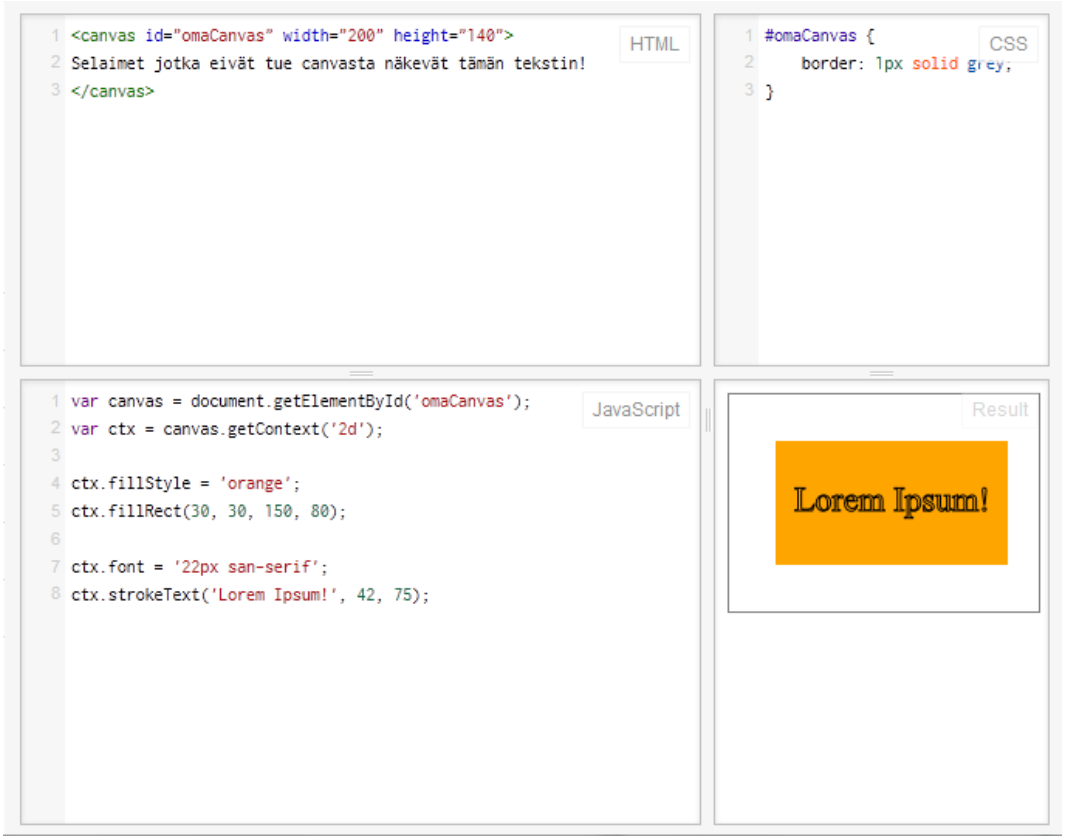
rect()-metodi luo suorakulmion canvasille. fillRect()-metodi piirtää täytetyn suorakulmion canvasille. strokeRect()-metodi piirtää suorakulmion, jonka ympärillä on reunus. clearRect()-metodi tyhjentää määritetyn suorakaiteen muotoisen alueen canvasista. Tätä metodia käytetään muun muassa silloin, kun canvas halutaan tyhjentää kokonaan. (W3Schools 2013a.)

fill()-metodia käytetään täyttämään kyseisellä hetkellä piirrossa oleva polku (path). stroke()-metodi käytännössä piirtää määritellyn polun, eli piirtää viivat. beginPath() metodi aloittaa polun. moveTo()-metodi siirtää kynää cavasilla määriteltyyn kohtaan ilman viivan piirtoa. closePath()-metodi luo polun kyseisestä kohdasta polun alkupisteeseen. lineTo()-metodi luo uuden pisteen ja luo viivan viimeeksi määriteltyyn pisteeseen. arc()-metodi luo kaaren (arc) / käyrän (curve). arcTo()-metodi luo kaaren/käyrän kahden tangetin välille. (W3Schools 2013a.)

scale()-metodi skaalaa piirrostsa pienemmäksi tai suuremmaksi. rotate()-metodi kääntää piirrosta haluttuun suuntaan. translate()-metodi uudelleen asettaa canvaksen koordinaatiston (0,0) kohdan. fillText()-metodi piirtää canvasille ”täytetyn” tekstin. strokeText()-metodi piirtää tekstin canvasille, mutta piirtää vain ääriviivat tekstille. drawImage()-metodi piirtää kuvan, toisen canvasin tai videon canvasille. toDataURL()-metodi palauttaa canvas-elementissä olevan kuvan merkkijonomuodossa. save()-metodi tallentaa canvasin nykyisen tilan. restore()-metodi palauttaa aiemmin tallennetun canvasin tilan ja ominaisuudet. (W3Schools 2013a.)

Ominaisuuksia, joita getContext()-metodin palauttama objekti sisältää, ovat muun muassa seuraavat: fillStyle, strokeStyle, shadowColor, lineCap, lineJoin, lineWidth, font, textAlign, width, height ja data. FillStyle-ominaisuus asettaa tai palauttaa värin, liukuvärin tai kuvion, mitä käytetään piirroksen täyttämiseen. StrokeStyle-ominaisuus asettaa tai palauttaa värin, liukuvärin tai kuvion, mitä käytetään piirtämiseen. ShadowColor-ominaisuus asettaa tai palauttaa varjoissa käytettävän värin. LineCap-ominaisuus asettaa tai palauttaa viivan päätyjen tyylin. LineJoin-ominaisuus asettaa tai palauttaa tyylin, mitä käytetään viivojen kulmakohdissa. LineWidth-ominaisuus asettaa tai palauttaa viivan paksuuden. Font-ominaisuus asettaa tai palauttaa teksteille käytetyn fontin. TextAlign-ominaisuus asettaa tai palauttaa tekstien ryhmittymisen. Width-ominaisuus palauttaa ImageData-objektin leveyden. Height-ominaisuus palauttaa ImageData-objektin korkeuden. Data-ominaisuus palauttaa objektin, mikä pitää sisällään määritelly ImageData-objektin tiedot. (W3Schools 2013a.)

2D-grafiikan lisäksi canvasiin on myös mahdollista luoda 3D-grafiikkaa WebGL-rajapinnan avulla. Kuviossa 8 nähdään esimerkki canvasin käytöstä ja sen luomasta grafiikasta. (Wikipedia 2013b.)



KUVIO 8. Canvas-esimerkki

Kuvion 8 HTML-laatikossa nähdään canvas-elementti, jolle on annettu uniikki id sekä leveys 200 pikseliä ja korkeus 140 pikseliä. Tagien sisällä esiintyvä teksti näytetään, jos käyttäjän internetselain ei ole yhteensopiva canvas-elementin kanssa. Oikealla ylhäällä määritellään 1 pikselin levyinen harmaa reunus canvasin ympärille, jotta canvasin rajat hahmottuisivat paremmin. JavaScript-laatikossa ensimmäisellä rivillä etsitään canvas uniikin id:n avulla. Kuvion 8 toisella rivillä oleva getContext(’2d’)-funktio palauttaa siis objektin, joka tarjoaa tarvittavat metodit ja ominaisuudet, jotta canvasille piirtäminen on mahdollista. Riveillä 4 ja 5 määritetään täyttöväri oranssiksi ja piirretään canvasille 30 pikselin päästä yläreunasta ja 30 pikselin päästä vasemmasta reunasta 150 pikseliä leveä ja 80 pikseliä korkea suorakulmio. Riveillä 7 ja 8 määritellään fontti kokoon 22 pikseliä ja fontiksi san-serif, minkä jälkeen kirjoitetaan teksti ”Lorem Ipsum!” annettuun kohtaan. Esimerkki on yksinkertainen, mutta antaa käsityksen siitä, miten canvas-elementtiin piirtäminen tapahtuu.

2.6 jQuery

jQuery on selainriippumaton JavaScript-kirjasto, mikä on pääasiassa suunniteltu helpottamaan selainpuolen ohjelmointia. jQuery on julkistettu tammikuussa 2006, ja nykyään yli 55 prosenttia maailman eniten vierailluista web-sivuista käyttää sitä. jQuery on tällä hetkellä kaikkein suosituin JavaScript-kirjasto. (Wikipedia 2013g.)

jQuery on ilmainen, perustuu avoimeen lähdekoodiin, ja se on lisensoitu MIT-lisenssin alaisuuteen. jQueryn syntaksi on tehty mahdollisiman helpoksi käyttää ja ymmärtää, minkä vuoksi se on suosittu. jQuery helpottaa dokumentissa navigointia, DOM-elementtien valitsemista ja jQueryllä on helppo toteuttaa erinäisiä animaatioita, tapahtumankäsittelijöitä sekä tehdä Ajax-pohjaisia ohjelmia. jQueryssä on myös tuki liitännäisille, joiden ansioista jQueryn ominaisuuksien laajentaminen on mahdollista. (Wikipedia 2013g.)

jQuery sisältää seuraavanlaisia ominaisuuksia:

- DOM-elementtien valinnat käyttäen selainriippumatonta avoimen lähdekoodin valitsinmoottoria nimenltä Sizzle
- DOM:n läpikäynti ja muuttaminen mukaan lukien tuki CSS1-3:lle
- tuki tapahtumille ja niiden käsittelijöille
- DOM-manipulaatio, joka perustuu CSS-valitsijoihin, jotka käyttävät elementin nimeä ja elementin atribuutteja (id ja class) rakentaakseen valitsijoita
- efektit ja animaatiot
- Ajax
- laajennettavuus plug-ineilla
- apuohjelmat, kuten käyttäjäagentin (user agent) ominaisuuksien tunnistus
- takautuva yhteensopivuus funktioille, kuten inArray() ja each(), jotka löytyvät natiiveina moderneista selaimista
- selaimesta riippumaton tuki (cross browser)

(Wikipedia 2013g).

jQueryn perussyntaksi on $(selektori).tapahtuma(), jossa $-merkki on lyhenne funktiolle nimeltä jQuery, (selektori) valitsee elementin johon tapahtuma() kohdentuu. Yksi jQueryn vahvuuksista on helpoiksi tehdyt css-valitsimet, joiden avulla elementtien valitseminen ja käyttäminen on helppoa. Esimerkkejä jQueryn CSS-valitsimista voidaan nähdä taulukossa 1, jossa on kattava katsaus erityyppisiin valitsijoihin.

TAULUKKO 1. Esimerkkejä jQueryn valitsijoista (W3Schools 2013d)

Example	Selects
$("*")	All elements
$("#lastname")	The element with id="lastname"
$(".intro")	All elements with class="intro"
$(".intro,.demo")	All elements with the class "intro" or "demo"
$("p")	All <p> elements
$("h1,div,p")	All <h1>, and <p> elements
$("p:first")	The first <p> element
$("p:last")	The last <p> element
$("tr:even")	All even <tr> elements
$("tr:odd")	All odd <tr> elements
$("p:first-child")	All <p> elements that are the first child of their parent
$("p:last-child")	All <p> elements that are the last child of their parent
$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent
$("div > p")	All <p> elements that are a direct child of a element
$("div p")	All <p> elements that are descendants of a element
$("div + p")	The <p> element that are next to each elements
$("div ~ p")	All <p> elements that are siblings of a element
$("input:not(:empty)")	All input elements that are not empty
$(":focus")	The element that currently has focus
$(":contains('Hello')")	All elements which contains the text "Hello"
$(":empty")	All elements that are empty
$("[href]")	All elements with a href attribute
$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
$(":input")	All input elements

Valitsijoilla voidaan siis valita elementti tai useita elementtejä muun muassa id:n, luokan ja elementin perusteella. Valitsijaan voidaan myös lisätä suodattimia, jotka yhdistetään valintaan kaksoispisteellä. Suodattimia ovat muun muassa first, last, odd ja even, joilla voidaan valita joko ensimmäinen tai viimeinen elementti tai

sitten parilliset tai parittomat elementit. Myös valitun elementin lapsielementteihin on mahdollista päästä käsiksi. Valitsijoita voidaan suodattaa elementille annettujen attribuuttien ja niille annetujen arvojen perusteella kuten esimerkiksi taulukossa 1 näkyvä ”[href=’default.htm’]”. (W3Schools 2013d.)

jQueryn sisältämät tapahtumat ja niiden käsittelijät ovat myös yksi hyvä syy käyttää jQueryä. Tapahtumalla tarkoitetaan esimerkiksi hiiren klikkausta, jonka seurauksena voidaan suorittaa haluttuja toimintoja. Tällä tavoin web-sovellukset saavat lisää dynaamisuutta. Yleisimpiä jQuery-tapahtumia ovat klikkaus, tuplaklikkaus, hiiren vieminen elementin päälle ja pois päältä, näppäimen painaminen, näppäimen pohjassa pitäminen ja ylös päästäminen, lomakkeen lähettäminen, lomakkeen arvon vaihtaminen, dokumentin tai ikkunan lataaminen, koon muuttaminen tai vierittäminen. Jokaiseen näistäkin tapahtumista voidaan lisätä toimintoja. (W3Schools 2013c.)

jQueryn käyttöönotto tapahtuu joko linkittämällä paikalliset kopiot jQuery-tiedostosta HTML-dokumentissa tai linkittämällä ne suoraan joltain julkiselta serveriltä esimerkiksi Googlen tai Microsoftin palvelimilta. Kuviossa 9 nähdään esimerkkejä jQueryn käyttöönotosta ja käytöstä. (Wikipedia 2013g.)

```
<!DOCTYPE html>
<html lang="fi">
<head>
    <meta charset="utf-8">
    <title>JQ-esim</title>
    <script type="text/javascript" src="jquery-1.9.1.min.js"></script>
    <script>
        $(document).ready(function(){

            $("#nappi1").click(function(){
                $(".piilossa").show();
                jQuery(".piilossa").css("color", "green");
            });

            $("p").each(function(i){
                $(this).html("nro "+ (i+1));
            });

        });
    </script>
</head>
<body>
    <div id="laatikko">
        <div id="nappi1">klikkaa!</div>
        <div class="piilossa" style="display:none;">Klikattu!</div>

        <p></p>
        <p></p>
        <p></p>
    </div>
</body>
</html>
```

klikkaa!

nro 1

nro 2

nro 3

KUVIO 9. jQueryn esimerkkikoodi ja linkitys

Kuviossa 9 on tavallinen HTML5-dokumentti, jossa rivillä kuusi tapahtuu jQueryn linkittäminen paikallisesta sijainnista. Rivillä kahdeksan löytyy ensimmäinen jQuery-funktio, jossa valitaan valitsijalla $(document), eli koko HTML-dokumentti ja pisteellä siihen ketjutetaan funktio ready(), jolle annetaan parametriksi anonyymifunktio. Käytännössä rivi kahdeksan siis odottaa, että HTML-dokumentti on täysin latautunut, ja sen jälkeen suorittaa sen sisällä olevan funktion.

Rivillä 10 on esimerkki siitä, miten jQueryllä voidaan tehdä klikkauksen kuuntelija. Kuuntelija sidotaan id:llä nappi1 olevaan div-elementtiin ja klikkauksen tapahtuessa suoritetaan parametriksi annettu funktio. Funktion sisällä voidaan nähdä kaksi tapaa, joilla jQueryä voidaan käyttää, eli $-merkki tai kirjoittamalla jQuery. Rivi 11 näyttää piilotetun div-elementin ja seuraava rivi 12 muuttaa dynaamisesti kyseisen divin arvon color arvoksi green.

Rivillä 15 alkavassa funktiossa ensin valitsijalla valitaan HTML-dokumentin p-elementit ja funktiolla each() käydään läpi jokainen löydetty elementti ja jokaiselle löydetylle elementille suoritetaan parametriksi asetettu funktio. Tämä funktio valitsee $(this)-valitsijalla juuri tämän kyseisen elementin, jonka each on löytänyt ja muuttaa sen p-elementin sisällöksi annetun arvon.

2.7 jQuery Ajax

Yhtenä jQueryn maininnan arvoisista ominaisuuksista voidaan pitää myös sen helppoa tapaa tehdä Ajax-kutsuja. Ajax tulee sanoista Asynchronous JavaScript And XML. Ajax on siis tekniikka, jolla voidaan lähettää ja vastaanottaa dataa serveriltä asynkronisesti taustalla. Paluu data, eli vastaus tulee Ajaxissa nykyään yleensä JSON muodossa ennen käytetyn XML:n sijaan. Ajax ei siis ole yksittäinen tekniikka, vaan siinä on yhdistelty monia teknologioita. HTML:ää ja CSS:ää käytetään sivuston rakenteen luomiseen ja JavaScripti hallitsee DOM:a ja sivuston dynaamisia muutoksia. JavaScript ja XMLHttpRequest-objekti mahdollistaa datan siirtämisen palvelimen ja selaimen välillä asynkronisesti ilman, että koko sivua tarvitsee päivittää. (Wikipedia 2013a.)

jQueryssä Ajaxia voidaan kutsua yksinkertaisesti funktiolla jQuery.ajax(url [,settings]); ensimmäinen parametri on String, eli merkkijono muotoinen ja se pitää sisällään sen osoitteen, mihin kyseinen kutsu tullaan lähetettämään. Toinen parametri on settings, eli asetukset, joka on vapaaehtoinen parametri, ja se on muotoa PlainObject, eli ”tavallinen objekti”. Se pitää sisällään joukon avain-arvo-pareja, joilla Ajax kutsua voidaan määritellä ja muokata halutunlaiseksi. Mahdollisia määrityksiä joita jQueryn Ajax-funktion settings-parametrille voidaan antaa ovat seuraavat: accepts, async, beforeSend, cache, complete, contents, contentType, context, converters, crossDomain, data, dataFilter, dataType, error, global, headers, ifModified, isLocal, jsonp, jsonpCallback, mimeType, password, processData, scriptCharset, statusCode, success, timeout, traditional, type, username, xhr ja xhrFields. Accepts-määrityksen tyyppi on PlainObject ja sen oletusarvo on riippuvainen DataType-määrityksestä. (Wikipedia 2013a.)

Accepts-määritys lähetetään request headerille, ja se kertoo serverille, millaisen vastauksen se hyväksyy. Async-määrityksen tyyppi on Boolean ja oletusarvo on true, eli totta. Oletuksena kaikki Ajax kutsut lähetetään asynkronisesti, mutta jos on tarvetta synkroniselle kutsulle, niin tämä määritys täytyy asettaa arvoksi false, eli epätosi. (Wikipedia 2013a.)

Asetettaessa async-määrtitystä epätodeksi täytyy muistaa, että kutsu saattaa väliaikaisesti lukitta internetselaimen, eikä sillä pystytä suorittamaan muita toimintoja, kun kutsu on aktiivisena. (Wikipedia 2013a.)

BeforeSend-määrityksen tyyppi on funktio Function(jqXHR jqXHR, PlainObject settings), missä parametreina ovat jqXHR, eli jQuery XMLHttpRequest-objekti ja PlainObject settings, mikä sisältää asetuksia. BeforeSend on siis käytännössä esikutsu, jota voidaan käyttää jqXHR-objektin muokkaamiseen ennen kuin se lähetetään. BeforeSendiä voidaan myös käyttää kustomoitujen otsikkotietojen määrittämiseen. (Wikipedia 2013a.)

Cache-määritys on tyyppiä Boolean, ja jos cache määritetään epätodeksi, niin internetselain ei tallenna välimuistiin haettuja sivuja. Täytyy myös muistaa, että cache-määrityksen false-arvo toimii oikein vain HEAD- ja GET-kutsuissa. (Wikipedia 2013a.)

Complete Ajax-määritys on tyyppiä funktio(jqXHR jqXHR, String textStatus), ja tätä funktiota kutsutaan silloin, kun Ajax-kutsu päättyy, eli heti kun success- ja error callback-funktiot on suoritettu. Complete-funktiolla on kaksi parametriä, jqXHR-objekti ja merkkijono, mikä kategorisoi kutsun tilan. Sen arvoja voivat olla success, notmodified, error, timeout, abort tai parseerror. (Wikipedia 2013a.)

Contents-määritys on tyyppiä PlainObject, ja tämän määrityksen tarkoitus on määrittää se, miten jQuery jäsentää sen sisältötyypille annetun vastauksen. Seuraava määritys on nimeltään contentType, jonka tyyppi on String, eli merkkijono, ja sen oletusarvo on 'application/x-www-form-urlencoded; charset=UTF-8'. ContentType-määrityksen oletusarvo on riittävä useimmissa tapauksissa ja sitä ei ole tarvetta vaihtaa, mutta harvoissa tapauksissa esimerkiksi charset-arvoa on tarvetta muuttaa. (Wikipedia 2013a.)

Converters-määritys on myös tyyppiä PlainObject ja tämä objekti pitää sisällään datatyyppien väliset muuntimet. Jokaisen muuntimen arvo on funktio, mikä palauttaa muunnetun arvon Ajax-vastaukseen. (Wikipedia 2013a.)

CrossDomain-määrityksen tyyppi on Boolean, ja jos se asetetaan arvoon tosi, niin se mahdollistaa crossDomain-kutsun samasta domainista. Silloin on esimerkiksi mahdollista tehdä serverin puolelta uudelleenohjaus toiseen domainiin. CrossDomain-määrityksen oletusarvo on false, jos kyseessä on same-domain-kysely, ja true, jos kyseessä on cross-domain-kysely. (Wikipedia 2013a.)

Data-määrityksen tyyppi on joko PlainObject tai String, eli merkkijono, ja tämä määritys sisältää sen datan, joka lähetetään internetselaimesta serverille. Data-määritys käännetään aina merkkijonoksi, jos se on muotoa PlainObject. Data liitetään url:ään GET-kutsuja silmällä pitäen, mutta tämänkin automaation voi estää myöhemmin esiteltävässä processData-määrityksen yhteydessä. (Wikipedia 2013a.)

DataFilter-määritys on tyyppiä Function(PlainObject data, String type)-function, joka palauttaa objektin. DataFilteriä voidaan käyttää prosessoimaan käsittelemätöntä XMLHttpRequest-dataa. DataFilteriä käytetään suodattamaan ja siistimään kutsuun tullut vastaus. DataFilter-funktio ottaa sisäänsä kaksi parametriä, joista ensimmäinen on raakadata, joka on tullut serveriltä, ja toisena on datan tyyppi. (Wikipedia 2013a.)

DataType-määritys on tyyppiä String, eli merkkijono ja sen oletusarvona on jQueryn oma Intelligent Guess, eli ”älykäs arvaus”-tekniikka, joka valitsee määrityksen arvoksi joko ”xml”, ”json”, ”script” tai ”html”. DataTypen arvoksi määritetään sen tyyppinen data, mitä odotetaan vastaukseksi serveriltä. (Wikipedia 2013a.)

Error-määritys on tyyppiä Function(jqXHR, String textStatus, String errorThrown)-funktio, ja tätä määriteltyä funktiota kutsutaan aina silloin, kun kyseinen Ajax-kysely epäonnistuu. Funktiossa tulee mukana kolme parametriä, joista ensimmäinen on jqXHR-objekti, toisena tulee merkkijono, joka kertoo, minkä tyyppinen virhe on kyseessä, ja tämän toisen parametrin mahdollisia arvoja

ovat ”timeout”, ”error”, ”abort” ja ”parsererror”. Aina kun http-virhe tapahtuu, niin kolmas parametri errorThrown sisältää merkkijonon, missä on selkokielinen selitys sattuneesta virheestä, kuten esimerkiksi ”Not Found” tai ”Internal Server Error”. Error-määritykseen voidaan myös laittaa arvoksi taulukko, joka sisältää useita funktioita, ja näistä funktioista jokaista kutsutaan vuorollaan. (Wikipedia 2013a.)

Global-määrityksen tyyppi on boolean ja oletusarvo true, eli tosi. Tällä määrityksellä on mahdollista hallita erinäisiä Ajax-tapahtumia ja sitä, suoritetaanko niitä vai ei. (Wikipedia 2013a.)

Headers-määritys on oletusarvoisesti tyhjä objekti, mutta voi sisältää avain-arvo-pareja, jotka lähetetään kyselyn mukana serverille. Headers-määrityksen sisältämät arvot asetetaan ennen kuin beforeSend-määrityksen funktiota kutsutaan ja täten Headers-määrityksen sisältämiä arvoja voidaan ylikirjoittaa beforeSend-funktiossa. (Wikipedia 2013a.)

IfModified-määritys on Boolean tyyppinen ja oletusarvoisesti false, eli epätosi. IfModified sallii Ajax-kyselyn olla onnistunut vain siinä tapauksessa, jos vastaus on muuttunut viime kyselystä. Tämä tapahtuu tarkistamalla Last-Modified-otsikkotieto ja vertaamalla sitä nykyiseen. (Wikipedia 2013a.)

IsLocal-määrityksen oletusarvo riippuu nykyisestä sijaintiprotokollasta ja on Boolean-tyyppinen. Tämä määritys sallii sen, että kyseinen ympäristö tunnistetaan ”paikalliseksi” ympäristöksi, vaikka näin todellisuudessa ei olisikaan. (Wikipedia 2013a.)

Jsonp-määritys on tyyppiä String, eli merkkijono, ja se yliajaa callback-fuktion jsonp-pyynnössä. JsonpCallback on jsonp-määritykseen liittyvä määritys, joka on tyyppiä String tai Function(), ja tämä määritys määrittää tapahtuvan callback-funktion Jsonp-pyynnössä. (Wikipedia 2013a.)

ProcessData-määrtys on Boolean-tyyppinen ja oletusarvona true, eli tosi. ProcessDatan ollessa tosi Data-määritykseltä tullut tieto käännetään aina merkkijonomuotoon, joka täsmää ContentType-määritykseen. Jos siis halutaan

lähettää joko DOM-dokumentti tai jokin muu ei-prosessoitu data eteenpäin, tulee tämä arvo asettaa epätodeksi. (Wikipedia 2013a.)

StatusCode-määritys on tyyppiä PlainObject ja sen sisään tulee numeroita, jotka vastaavat http:n numeerisia koodeja, ja niille pariksi laitetaan funktio, joka suoritetaan, jos vastauksena tulee kyseinen koodi. Esimerkiksi jos Ajax-kyselyn vastauksena tulee koodi 404, voidaan kyseiseen numeroon liittää vaikka funktio, joka ilmoittaa asiasta käyttäjälle. (Wikipedia 2013a.)

Succes on hyvin usein käytetty määritys, ja se on tyyppiä Function(PlainObject data, String textStatus, jqXHR). Tätä success-kohdassa määriteltyä funktiota kutsutaan aina silloin, kun Ajax-kutsu onnistuu. Funktiossa tulee mukana kolme parametriä, joista kaksi viimeisintä on jo ennaltaan tutut textStatus- ja jqXHR-objekti, mutta ensimmäinen parametri sisältää sen datan, jonka serveri on tehdyn Ajax-kutsun toimesta palauttanut takaisin. Tässä funktiossa voidaan käsitellä palautettu data ja vaikka dynaamisesti näyttää se käyttäjälle. (Wikipedia 2013a.)

Timeout-määritykselle annetaan arvoksi numero millisekunteina, jonka kuluessa kyseinen Ajax-kutsu tulee suorittaa tai muuten kutsu muuttuu virheelliseksi. (Wikipedia 2013a.)

Type-määritys on tyyppiä String, ja sen oletusarvo on ”GET”. Tämä määrittää sen, minkä tyyppinen pyyntö palvelimelle tehdään. Mahdollisia Type-määrityksen arvoja ovat esimerkiksi ”GET”, ”POST”, sekä ”PUT” ja ”DELETE”, mutta kaikki internetselaimet eivät tue ”PUT”- ja ”DELETE”-arvoja. (Wikipedia 2013a.)

Kaikki esiteltyjä arvoja ei ole pakko, eikä kannata laittaa jokaisen Ajax-kutsun yhteyteen, vaan ne voidaan myös asettaa niin sanoituiksi oletusarvoiksi jQueryn sisällä olevan funktion $.ajaxSetup() avulla. Kuviossa 10 nähdään esimerkki yksinkertaisesta ja helposta Ajax-kutsusta. (Wikipedia 2013a.)

jQueryAjaxExample.js

```
1   jQuery.ajax({
2       type: "POST",
3       url: "myphpthings.php",
4       data: {
5           "name":"Matti Meikäläinen",
6           "age":"47",
7           "sex":"male"
8       },
9       dataType: "json",
10      timeout: 2000,
11      success: function(data) {
12          $("#mydiv").append(data);
13      },
14      error: function() {
15          alert("Something went terribly wrong here!");
16      },
17      complete: function() {
18          $("#theend").dialog();
19      }
20  });
```

KUVIO 10. Ajax esimerkki

Kuvion 10 esimerkin ensimmäisellä rivillä suoritetaan jQueryn Ajax-funktio, jolle asetuksina annetaan aiemmin esiteltyjä määrityksiä. Rivillä kaksi määritettään Ajax-kutsun tyypiksi ”POST”. Rivillä kolme määritellään osoite, johon kutsu tullaan lähettämään, joka on tässä tapauksessa myphpthings.php. Lähetettäväksi dataksi määritellään objekti, joka sisältää kohdan ”name” ja sen arvon ”Matti Meikäläinen”, kohdan ”age” ja sen arvon ”47”, sekä kohdan ”sex” ja sen arvon ”male”. Rivillä yhdeksän määritellään dataTypen arvoksi ”json”, eli kerrotaan, että paluudata halutaan json-muodossa. Rivillä 10 määritellään timeout-määrityksen arvoksi 2000 millisekuntia, eli kutsu keskeytetään virheellisenä, jos vastausta ei kahteen sekuntiin saada. Riviltä 11 alkaa success-funktio, jonka ainoana parametrinä tulee serverin palauttama data ja rivillä 12 palautettu data liitetään id:llä ”mydid” olevan divin sisään jQueryn append()-funktiolla. Rivillä

14 alkaa error-funktio, eli mikäli kutsussa tapahtuu virhe, niin käyttäjälle näytetään varoitus, missä lukee teksti ”Something went terribly wrong here!”. Success- tai error-funktion suorituksen jälkeen suoritetaan riviltä 17 alkava complete-funktio, mikä avaa id:llä ”theend” olevan dialogin ja näyttää sen käyttäjälle. Tämä oli vain yksinkertainen esimerkki siitä, miten helposti Ajax-kutsun voi tehdä käyttämällä jQueryä.

2.8 JSON

JSON tulee sanoista JavaScript Object Notation, ja se on yksinkertainen tiedonsiirtomuoto. JSONia on helppo lukea ja kirjoittaa sen yksinkertaisen rakenteensa johdosta. JSON ei nimen harhaanjohtavuudesta huolimatta ole pelkästää JavaScriptiä varten, vaan sitä voidaan käyttää myös muissa ohjelmointikielissä, kuten C:ssä, C++:ssa, C#:Ssa, Javassa, Perlissä, Pythonissa ja PHP:ssa. (JSON.org 2013.)

JSON koostuu kahdesta rakenteesta, kokoelmasta avain-arvo-pareja ja järjestetystä arvoja sisältävästä listasta, eli taulukosta. Nämä rakenteet ovat universaaleja datarakenteita, ja vastaavanlaisia rakenteita löytyy sisäänrakennettuna monesta ohjelmointikielestä, eli jokainen moderni ohjelmointikieli tukee näitä rakenteita tavalla tai toisella. (JSON.org 2013.)

JSON-objekti alkaa vasemmalla kaarisulkeella, eli ”{”-merkillä ja loppuu oikeaan kaarisulkeeseen, eli ”}”-merkkiin. Objektissa olevaan avaimeen liitetään arvo kaksoispisteellä ja avain-arvo-parit erotellaan pilkuilla, eli rakenteesta tulee esimerkiksi ”{avain1:arvo1, avain2:arvo2}”. Taulukko on järjestetty kokoelma arvoista, ja arvo alkaa aina vasemmalla hakasulkeella, eli ”[”-merkillä ja loppuu oikeaan hakasulkeeseen, eli ”]”-merkkiin. Taulukon arvot erotellaan toisistaan pilkuilla, eli taulukon rakenne on esimerkiksi ”[1,5,2,0]”. Objektin tai taulukon arvo voi olla merkkijono, numero, true, eli tosi, false, epätosi, null, toinen objekti tai toinen taulukko, eli objekteja tai taulukoita voi olla objektin tai taulukon sisällä. (JSON.org. 2013.)

Yleisin tapa käyttää JSONia on Ajax-kutsujen yhteydessä, jossa lähettää objekteja JavaScriptiltä PHP:lle ja molemmat saavat datan ymmärrettävässä muodossa.

Kuviossa 11 on esitetty yksinkertainen esimerkki JSON:sta ja sen helppolukuisuudesta.

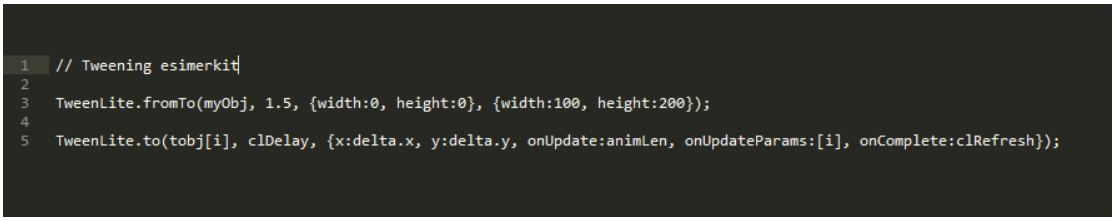
KUVIO 11. JSON-esimerkki (Wikipedia 2013h)

2.9 TweenLite JS

TweenLite JS on nopea, kevyt ja joustava tweenaukseen tarkoitettu JavaScript-kirjasto, joka mahdollistaa mm. erinäisten animaatioiden tekemisen. TweenLite on alun perin Flash alustalle toteutettu erillinen kirjasto, joka on mahdollistanut erilaisten objektien arvojen muuttamisen liukuen tietyn ajan kuluessa, mutta nyt se on myös saatavilla JavaScript-kirjastona. TweenLiten oma animointimoottori toimii jopa 20 kertaa nopeammin kuin jQueryn vastaava. TweenLiteä on käytetty tässä opinnäytetyössä sen nopeuden ja joustavuuden takia ja sillä on toteutettu lähes kaikki animaatiot, joita sovelluksessa on, kuten profiilien pituuksien muutoksissa olevat animaatiot. (Greensock.com 2013.)

TweenLitellä on mahdollista Tweenata, eli liu’uttaa mitä tahansa numeerista arvoa miltä tahansa objektilta ja TweenLite tukee myös monia erilaisia easingejä animaatioita varten. TweenLite on jQueryn tapaan myös laajennettavissa

liitännäisillä. TweenLitellä luotuja animaatiota on mahdollista toistaa, keskeyttää, aloittaa alusta ja jopa peruuttaa haluamallaan tavalla. (Greensock.com 2013.)



```
1  // Tweening esimerkit
2
3  TweenLite.fromTo(myObj, 1.5, {width:0, height:0}, {width:100, height:200});
4
5  TweenLite.to(tobj[i], clDelay, {x:delta.x, y:delta.y, onUpdate:animLen, onUpdateParams:[i], onComplete:clRefresh});
```

KUVIO 12. TweenLite esimerkki funktiot

TweenLiten käyttöönotto tapahtuu samaan tapaan kuin jQueryssä, eli linkittämällä paikallinen TweenLite.js-kirjasto HTML-dokumenttiin. Kuviossa 12 on kaksi esimerkkiä TweenLiten käytöstä. Rivillä neljä on perus TweenLite-funktio, joka käyttää fromTo()-funktiota hyväkseen, ensimäinen parametri on objekti, jonka arvoja halutaan tweenata, toinen parametri kertoo tweenaukseen käytetyn ajan, ja kolmas parametri on aaltosulkeiden sisällä ja kertoo tweenattavien arvojen lähtöarvot, neljäs ja viimeinen parametri kertoo aaltosulkeiden sisällä loppuarvot mihin tweenaus päättyy. Kaikessa yksinkertaisuudessaa rivi kolme tweenaa 1.5 sekunnissa myObj:n arvo leveys (width) nollasta sataan ja arvo pituus (height) nollasta kahteen sataan.

Rivillä viisi oleva tween-esimerkki on otos opinnäytetyön lähdekoodista, joka on hieman monimutkaisempi kuin ensimmäinen esimerkki. Tässä tweenissä objektin tobj[i] arvoja delta.x ja delta.y muutetaan clDelay-arvon määräämän ajan ja tweenin aikana kutsutaan funktiota animLen lukuisia kertoja, mikä piirtää canvas-elementtiin tapahtuvat muutokset. Lopulta, kun tweenaus on valmis, kutsutaan clRefresh-funktiota, mikä toimii tässä tapauksessa niin sanottuna callbackinä ja hoitaa tweenauksen päättymiseen liittyviä asioita.

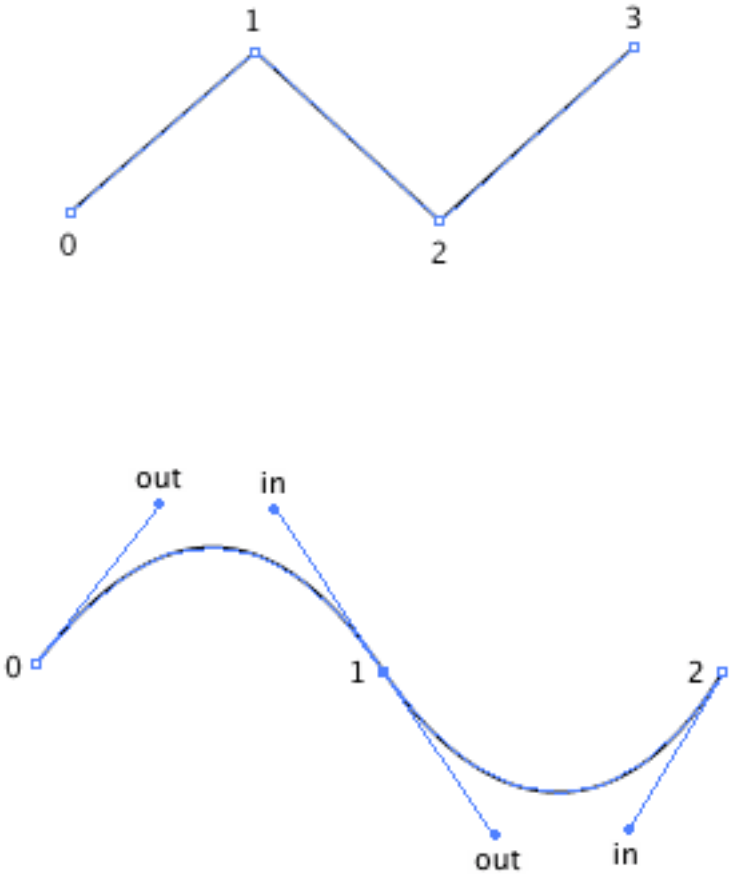
2.10 Paper.js

Paper.js on vapaaseen lähdekoodiin perustuva vektorigrafiikan ohjelmointikehys, joka toimii yhteen HTML5 canvas-elementin kanssa. Paper.js sisältää tehokkaita toiminnallisuuksia millä luodaan vektorigrafiikkaa ja bezier käyriä. Paper.js perustuu ja on pitkälti yhteensopiva Scriptographer, skriptaus ympäristön kanssa,

mikä on Adobe Illustratoria varten kehitetty ympäristö. Paper.js:n ovat kehittäneet Jürg Lehni ja Jonathan Puckey ja se on lisensoitu MIT-lisenssin alaisuuteen. (Paperjs.org 2013a.)

Paper.js tarjoaa kehittäjälleen helppokäyttöisen DOMin, eli Document Object Modelin. Paper.js:ssä voidaan luoda projekti, jonka sisään voi luoda kerroksia (layer), ryhmiä (group), polkuja (path) sekä rastereita. Ryhmät sekä kerrokset voivat sisältää muita kappaleita ja ryhmiä. Paper.js:ssä on globaali muuttuja ”document”, mikä viittaa aktiivisena olevaan koko Paper.js-dokumenttiin. Jokainen dokumentti sisältää joukon kerroksia. Kerroksiin päästään käsiksi document-objektin kautta, joko viittamalla nimeen tai indeksiin. Kerroksia voi olla yhdessä dokumentissa useita, ja ne piirtyvät canvasille järjestyksessä alhaalta ylöspäin. Kerroksien järjestystä voidaan muuttaa jälkikäteen. Kerroksien sisään voidaan luoda useita erilaisia polkuja ja kuvioita, joita voidaan jakaa ryhmiin kerroksien sisällä. Paper.js dokumenttia voi ajatella Adobe Photoshop projektin tyylisesti alustana, jossa on useita kerroksia ja kukin kerros sisältää grafiikkaa. (Scriptographer.org 2013a.)

Kerroksien sisältä löytyvät polut koostuvat segmenteistä. Polkuun lisättyjä segmenttejä voidaan helposti liikutella ja manipuloida. Segmentti liitetään toisiinsa kaarilla, ja ne koostuvat pisteestä ja kahdesta kahvasta (handle), jotka määrittelevät kaarien paikan ja suunnan. Segmentit, joille ei määritellä sankoja yhdistyvät suorin viivoin. Kuviossa 13 nähdään esimerkit kahdesta erilaisesta polusta, joissa nähdään segmentit numeroituina. (Scriptographer.org 2013d)

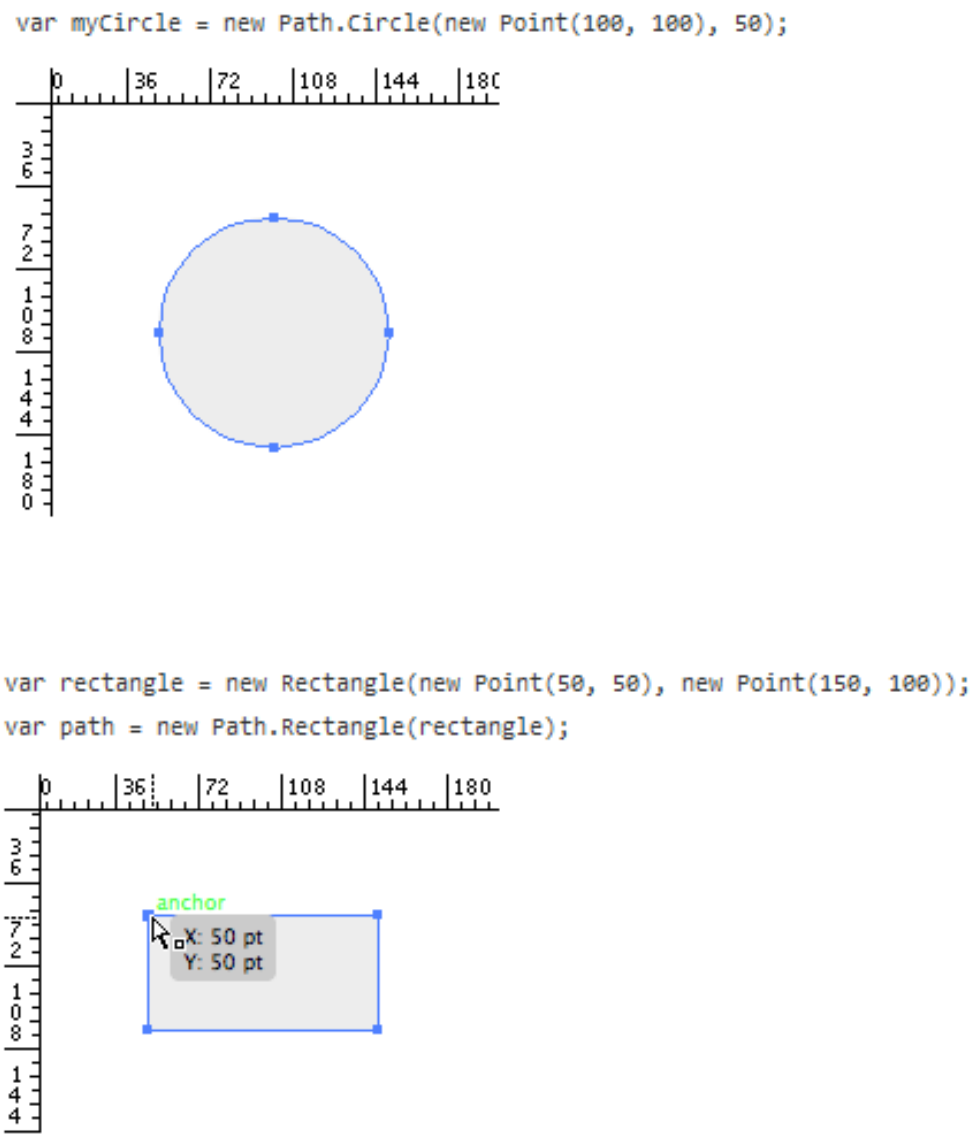


KUVIO 13. Esimerkkipolut numeroiduilla segmenteillä (Scriptographer.org 2013d)

Polkuja voidaan myös siloittaa kutsumalla myPath.smooth()-funtiota, jolloin Paper.js laskee polun segmenteille optimaalisimmat sankojen arvot ja kuvion 13 ylemmästä polusta saadaa alemman polun näköinen polku. Polku on myös mahdollista sulkea asettamalla myPath.closed-muuttujan arvoksi true, jolloin polun viimeisen ja ensimmäisen pisteen välille piirtyy myös kaarre. Polusta on myös mahdollista poistaa segmenttejä myPath.remove(index)-funktiolla, jolle määritetään parametriksi halutun segmentin indeksi. (Scriptographer.org 2013d.)

Erilaisia muotoja, kuten ympyröitä, suorakulmioita, kolmioita yms. on myös mahdollista luoda Paper.js:ssä helposti. Jokainen muoto on oma polkunsa.

Kuviossa 14 on esitetty esimerkkejä kuinka Paper.js:ssä luodaan erilaisia muotoja. (Scriptographer.org 2013d.)



KUVIO 14. Ympyrä ja suorakaide -esimerkki (Scriptographer.org 2013d)

Ympyrä luodaan yksinkertaisesti uutena polkuna Paper.js:stä valmiina löytyvällä Circle()-funktiolla, jolle annetaan parametreinä ympyrän keskipiste Point-objekti, sekä ympyrän säde pikseleinä. Suorakulmion luonti tapahtuu myös uutena

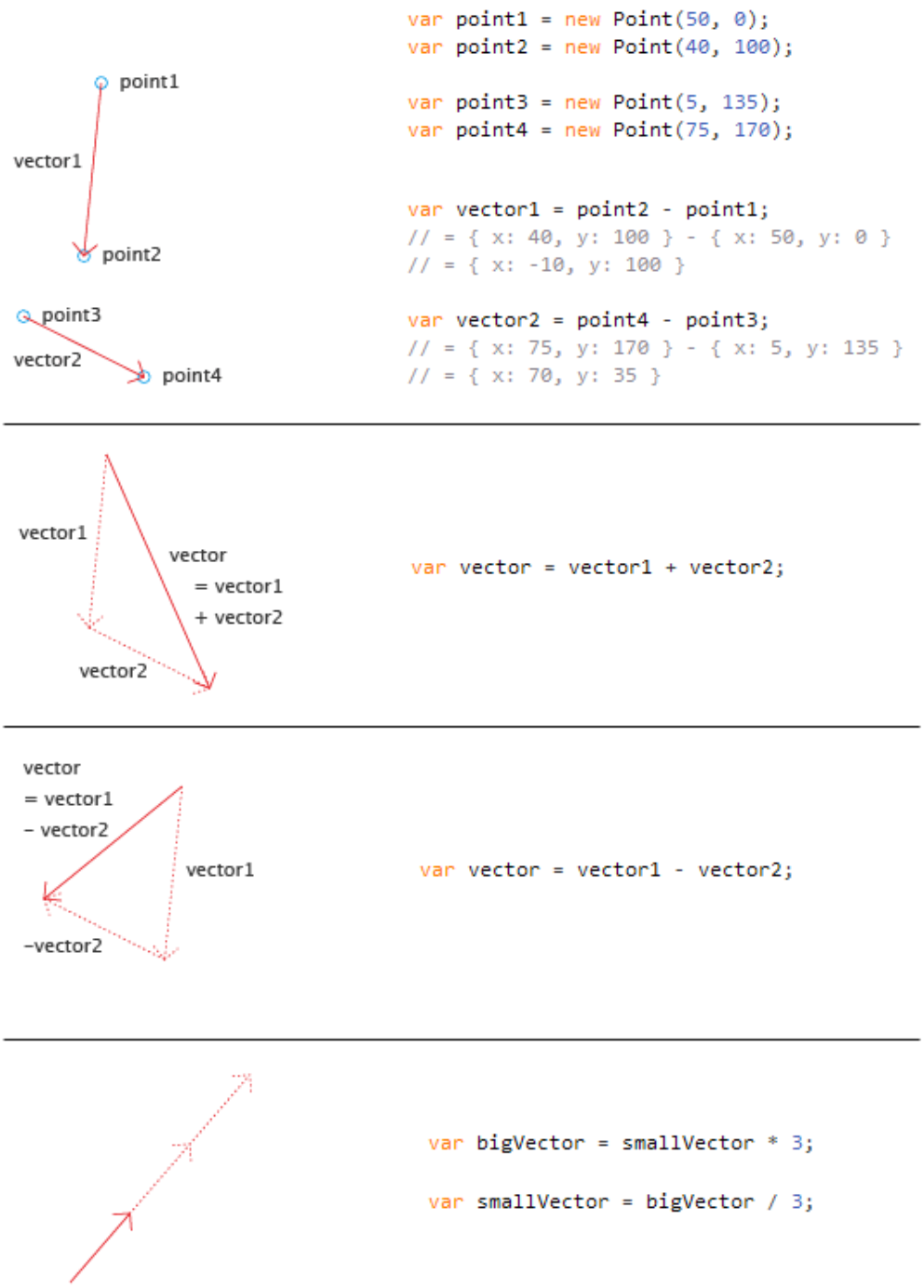
polkuna valmiista sisäänrakennetusta funktiosta. Suorakulmiolle täytyy määrittää alku- ja loppupiste. (Scriptographer.org 2013d.)

Paper.js:ään on myös sisään rakennettu hiiren tapahtumankäsittelijät erilaisille toiminnoille. Tällaisia tapahtumia on neljä kappaletta, joista ensimmäinen on onMouseDown(event). onMouseDown(event)-tapahtumankäsittelijää kutsutaan aina, kun käyttäjä painaa hiiren painiketta canvasin päällä. Toisen käsittelijän nimi on onMouseDrag(event), jota nimensä mukaan kutsutaan silloin, kun käyttäjä on painanut hiiren napin pohjaan ja liikuttaa hiirtä canvasin yllä. Kolmas ja viimeinen hiiren tapahtumankäsittelijä on onMouseUp(event), jota kutsutaan aina silloin, kun käyttäjä päästää hiiren napin ylös. Neljäs ja viimeinen hiiren tapahtumankäsittelijä on onMouseMove(event), jota kutsutaan silloin, kun käyttäjä liikuttaa hiirtä canvasin yllä. Jokaisessa hiiren tapahtumankäsittelijässä välitetään parametrinä ToolEvent-objekti, joka pitää sisällään tapahtumasta riippuen tietoa kyseisestä tapahtumasta. ToolEvent-objekti pitää sisällään seuraavanlaisia arvoja: point, lastPoint, downPoint, middlePoint, delta, pressure, count, type, item ja modifiers. Näiden arvojen avulla ja hiirenkäsittelijöitä hyväksi käyttäen mahdollistetaan muun muassa canvasille dynaaminen piirtäminen. (Scriptographer.org 2013b.)

Hiiren tapahtumankäsittelijöiden lisäksi Paper.js sisältää näppäintapahtumien käsittelijöitä kaksi kappaletta, joista ensimmäinen on onKeyDown(event), jota kutsutaan silloin, kun käyttäjä painaa näppäimistöltään näppäintä. onKeyUp(event)-käsittelijää taas kutsutaan silloin, kun käyttäjä päästää näppäimen ylös. Kumpaankin näppäinkuuntelijaa kuuluu hiiren tapaan oma KeyEvent-objekti, joka pitää sisällään arvoja, kuten type, character ja key. Type kertoo, millainen tapahtuma (event) on kyseessä, eli joko näppäimen ylös päästäminen (keyup) tai näppäimen painaminen (keydown)-tapahtuma. Character arvo kertoo painetun tai painettujen merkkien arvot. Key-arvo kertoo sen näppäimen, mitä on painettu. (Paperjs.org 2013b.)

Paper.js on, kuten aiemmin jo sanottu, vektorigrafiikan skriptauskehys, eli Paper.js:llä piirtäminen perustuu muun muassa vektoreihin. Paper.js:ssä vektori koostuu kahdesta point-objektista ja jokaisella point-objektilla on oma x- ja y-

koordinaatti. Vektori piirtyy siis näiden kahden pisteen välille, ja vektorilla itsellään on alkupiste x- ja y-koordinaattina sekä kulma eli suunta, mihin vektori piirtyy ja pituus. Vektoreita on mahdollista esimerkiksi laskea yhteen ja vähentää toisiinsa, minkä tuloksena syntyy uusi vektori, ja kertoa taikka jakaa luvuilla, jolloin vektorin pituus muuttuu. Paper.js:ään on myös sisäänrakennettu vektoreita varten muita monimutkaisempia matemaattisia funktioita, kuten dot(), cross() ja project(). Kuviossa 15 nähdään esimerkkejä vektorien luonnista, käytöstä ja niiden matemaattisista ominaisuuksista. (Scriptographer.org 2013c.)




```
var point1 = new Point(50, 0);
var point2 = new Point(40, 100);

var point3 = new Point(5, 135);
var point4 = new Point(75, 170);

var vector1 = point2 - point1;
// = { x: 40, y: 100 } - { x: 50, y: 0 }
// = { x: -10, y: 100 }

var vector2 = point4 - point3;
// = { x: 75, y: 170 } - { x: 5, y: 135 }
// = { x: 70, y: 35 }
```

```
var vector = vector1 + vector2;
```

```
var vector = vector1 - vector2;
```

```
var bigVector = smallVector * 3;

var smallVector = bigVector / 3;
```

KUVIO 15. Vektorit ja niiden ominaisuuksia (Scriptographer.org 2013c)

Kuviossa 15 nähdään siis osa vektorien matemaattisista ominaisuuksista. Kuvion 15 ylimmässä osassa luodaan ensimmäiseksi neljä kappaletta piste (point)-objekteja, joista tämän jälkeen luodaan kaksi vektoria. Vektorit luodaan vähentämällä loppupisteestä vektorin alkupiste. Toiseksi ylimmäisessä osassa on esimerkki siitä, miten vektoreita voidaan laskea yhteen. Toiseksi viimeisessä

osassa nähdään vektorin vähennyslasku, jolloin jälkimmäinen vektori on vastakkaissuuntainen. Kuvion 15 viimeisessä osassa nähdään esimerkki vektorin kertomisesta ja jakamisesta. Vektorin kertominen kasvattaa vektorin pituutta, kun taas jakaminen lyhentää sitä. Vektoreita on myös tämän lisäksi mahdollista muun muassa kääntää muuttamalla vektorin angle-arvoa. Vektorit siis mahdollistavat Paper.js:ssä monipuolisen tavan hallita polun pisteitä ja vektoreiden avulla piirtäminen on helppoa. (Scriptographer.org 2013c.)

3 SOVELLUKSEN TOTEUTUS, TOIMINTA JA OMINAISUUDET

Graafinen suunnitelutyökalu on tarkoitettu peltiprofiilien piirtämiseen ja mitoittamiseen. Työkalu on toteutettu pääosin teoriaosuudessa esiteltyjä HTML-, CSS- ja JavaScript-tekniikoita hyväksikäyttäen. Pohjana työkalussa toimii HTML:n canvas-elementti, jonka piirtämisen toteuttaa JavaScript-kirjasto Paper.js. Työkalu tuottaa vektorigrafiikkaa, eli piirroksia on mahdollisuus skaalailla mielin määrin. Työkalun toiminta perustuu pääosin kahteen osuuteen, piirto-osuuteen ja mitoitus-osuuteen. Työkalussa on myös mahdollista ladata jo piirrettyjä tai valmiiksi luotuja profiilimalleja, joita pystytään tämän jälkeen myös käsittelemään. Graafisen suunnittelutyökalun yleisilme ja grafiikat ovat Imager Oy:n suunnittelemia.

3.1 Sovelluksen piirto-osuus

Graafisen suunnitelutyökalun piirto-osuuden tarkoituksena on luoda alustava luonnos profiilin ulkonäöstä, joka tullaan myöhemmässä vaiheessa mitoittamaan. Jos työkalusta ladataan valmispohja tai jo olemassa oleva vanha tallennettu profiilimalli, ohjataan käyttäjä aina ensin piirto-osuuteen, mikäli käyttäjä haluaa tehdä muutoksia siihen. Kaikki canvasille tapahtuva piirto tehdään JavaSciptistä käsin, ja skripti on kirjoitettu <script> tagien sisään, jonka attribuutin type-arvoksi on annettu ”text/paperscript”, jotta tiedetään, että kyseinen skripti koskee Paper.js Javascript-kirjastoa sekä attribuutin canvas arvoksi asetetaan HTML-dokumentissa olevan canvas piirtoalustan uniikki id.

3.1.1 Piirtämisen aloittaminen

Piirtäminen aloitetaan klikkaamalla hiirellä paperia muistuttavaa canvas pohjaa. Canvas on tässä tapauksessa 990 pikseliä leveä ja 420 pikseliä korkea, mutta sen mitat on mahdollista määrittää uudelleen ja sovellus osaa itse sopeutua uusii mittoihin. Piirretyt pisteet sijoittuvat aina paperissa olevien ruudukoiden leikkauskohtiin, eli canvasiin on toteutettu ohjelmallisesti piirretty ruudukko, joille pisteet on sijoitettava, yhden ruudun koko on tässä esimerkissä 10 kertaa 10 pikseliä ja ruudukonkin mitat ovat muutettavissa. Kun ensimmäinen piste on

piirretty, voidaan piirtämistä jatkaa joko klikkaamalla uusi piste paperille tai pitämällä hiiren näppäintä pohjassa ja liikuttaa seuraava piste haluttuun kohtaan, jolloin uusi viiva on nähtävissä koko piirron ajan. Sovellus tallentaa kaikki piirretyt pisteet Paper.js:n Path, eli polku-objektiin ja jokainen piste on oma segmenttinsä. Segmentin pisteiden avulla määritetään piirtopinnalla olevat vektorit, joiden avulla sovellus piirtää oranssit viivat näkyviin. Kuviossa 16 nähdään esimerkki suunnitelutyökalulla piirretystä kuviosta.

KUVIO 16. Suunnittelutyökalun esimerkkipiirros

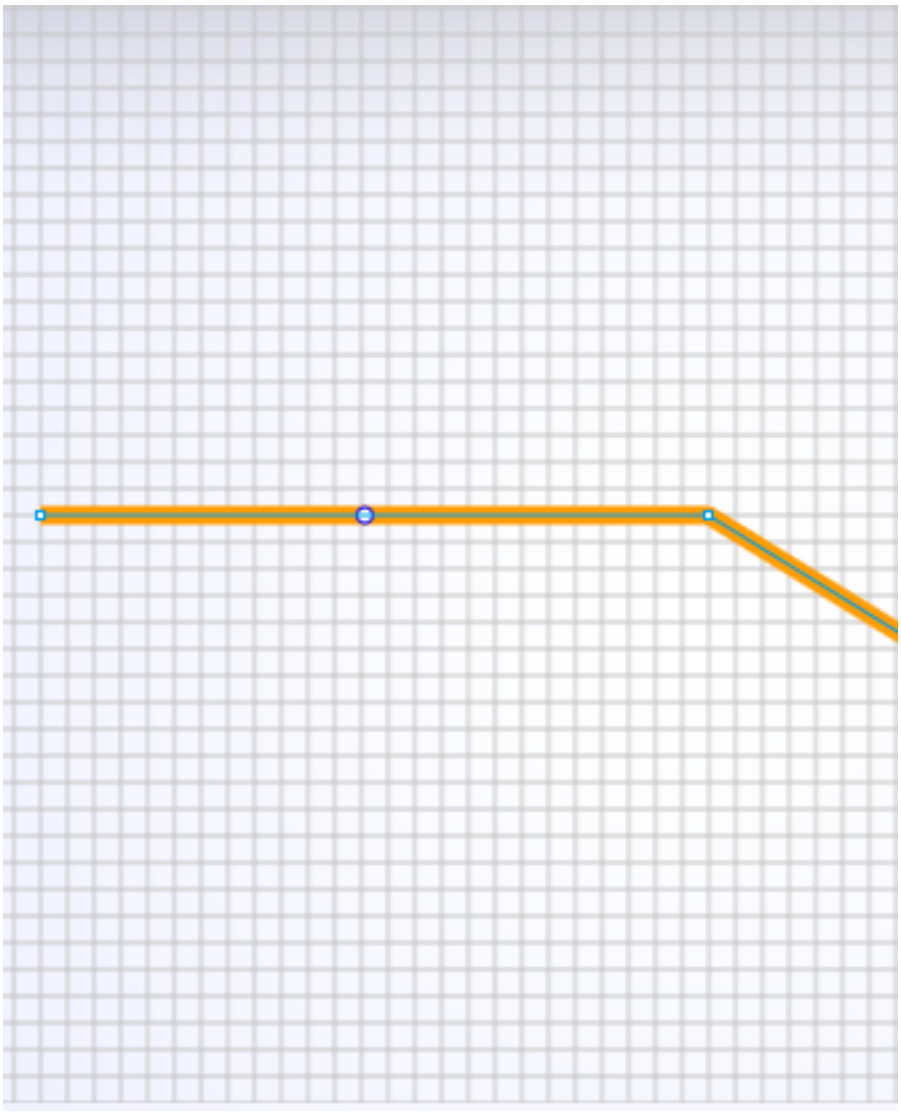
3.1.2 Pisteen paikan muuttaminen

Kuviossa 16 nähdään esimerkki siitä, miltä piirretty profiili näyttää. Piirretyn oranssin profiilin sisällä kulkee sininen viiva. Profiilin taitoskohdissa nähdään valkoisia palloja sinisellä reunuksella, eli segmenttejä, kun käyttäjä vie hiiren segmentin päälle voi hän halutessaa muuttaa pisteen paikkaa minne tahansa paperilla. Tällä tavoin käyttäjen ei tarvitse aina poistaa kuvion pisteitä alusta loppuun halutessaan siirtää yhtä pistettä. Paper.js:n polku-objektissa on hiiren tapahtuman kuuntelija, joka tarkkailee, onko hiiri kyseisen polku-objektin segmentin päällä. Hiiren ollessa tietyn liitoskohdan päällä kyseisen segmentin päälle piirretään suurempi valkoinen pallo oranssilla reunuksella ilmoittamaan osumasta. Oranssireunuksisen pallon ollessa näkyvissä segmentti on

liikutettavissa. Kyseinen pallo voidaan nähdä kuvion 16 toisen taitoskohdan kohdalla.

3.1.3 Pisteen lisääminen viivalle

Mikäli käyttäjä haluaa lisätä taitoksen keskelle jo olemassa olevaa viivaa, on sekin mahdollista. Halutessaan käyttäjä voi viedä hiiren kursorin viivan päälle, minkä seurauksena viivaan ilmestyy uusi valkoinen pallo kuvaamaan, että kyseiseen viivan kohtaan on mahdolista lisätä taite. Tässäkin tapauksessa polku-objektissa oleva tapahtuman kuuntelija tarkkailee hiirtä, mutta tällä kertaa ilmoittaa, jos hiiri on itse polun eikä segmentin päällä. Kuviossa 17 nähdään esimerkki, miltä taitoksen lisäämistä kuvaava pallo näyttää.



KUVIO 17. Taitoksen lisäämisen mahdollisuutta ilmoittava pallo

Kuviossa 17 piirretyn profiilin ensimmäisen viivan keskellä nähdään valkoinen pallo, mikä siis indikoi sitä, että kyseiseen kohtaan on mahdollista lisätä taite. Kun hiiren kursori on pallon päällä ja palloa aletaan hiirellä vetää esimerkiksi ylöspäin, siihen muodostuu uusi taite. Tässä tapauksessa polku-objektiin luodaan vain uusi segmentti hiiren osoittamaan kohtaan ja tietysti oikeaan väliin objektissa. Virheen sattuessa piirretyn profiilin pisteitä voidaan myös halutessaan tuhota aloittaen viimeisestä pisteetä aina profiilin alkuun saakka. Viimeisen segmentin tuhoaminen tapahtuu d-näppäintä painamalla. Heti kun näppäin tapahtuman kuuntelija havaitsee painalluksen näppäimistöllä, sovellus tarkistaa, onko kyseessä d-näppäin, ja jos näin on, niin polun viimeinen segmentti poistetaan.

Kuviosta 18 nähdään siisti lopputulos keskelle olemassa olevaa viivaa lisätystä taitteesta sekä liikutetusta profiilin päädystä.

KUVIO 18. Muokattu profiili, sekä lähennys, loitonnus ja keskitys toiminnot

3.1.4 Lähennys, loitonnus ja keskitys

Kuvion 18 oikeassa yläkulmassa voidaan nähdä kolme painiketta, jotka oikealta vasemalle ovat lähennys-, loitonnus- ja keskitys-toiminnot. Lähennys-painiketta painettaessa profiilin koko kasvaa ja loitonnus-painiketta painettaessa taas päinvastoin profiilin koko pienenee. Kun painiketta on painettu, sovellus skaalaa polun koon scale()-funktiota käyttäen ja TweenLite JavaScript -kirjasto animoi sulavasti profiilin koon muutoksen uuteen kokoon. Näitä toimintoja voidaan tarvita esimerkiksi jos johonkin ahtaaseen väliin tarvitsee lisätä uusi taitekohta tai taitekohtia on vierekkäin monia ja jotain niistä tarvitsee siirtää. Viimeinen, eli keskitä-painike on myös hyvin käytännöllinen; sitä painettaessa profiili keskitetään ja mahdutetaan kokonaan näkyviin olemassa olevalle paperille niin suureen kokoon kuin mahdollista. Kuviossa 19 nähdään, miltä piirustus näyttää, kun, keskitä-toimintoa on käytetty.

KUVIO 19. Keskitetty profiili

Profiilin piirto-osuuden ollessa valmis voidaan siirtyä mitoitus-osuuteen. Mitoitus-osuuteen päästää painamalla kuviossa 19 näkyvää ”Mitoita piirros” -painiketta.

3.2 Sovelluksen mitoitus osuus

Piirto osuuden jälkeen on siis edessä mitoitus-osuus. Mitoitus-osuuden tarkoituksena on antaa piirrokselle tarkat millimetrimitat sekä kulmat. Mitoitus osuuteen siirryttäessä sovellus tarkistaa kaikki rajoitukset, jotka profiilin taittokone on järjestelmälle antanut. Ensimmäisen lähtökulman arvo voi olla mikä tahansa, mutta seuraavien taitettujen kulmien arvon täytyy olla väliltä 45 – 315 astetta. Sovellus korjaa kaikki alittavat ja ylittävät arvot sallittuihin ääriarvoihin. Myös jokaisen viivan pituus tullaan tarkistamaan. Ensimmäisen ja viimeisen viivan pituus tulee olla yli 6 millimetriä ja näiden välissä olevien viivojen pituuden tulee olla enemmän kuin 15 millimetriä. Sovellus korjaa kaikki sallittuja lyhyemmät pituudet sallittuihin minimipituuksiin. Kuviosta 20 nähdään yleiskatsaus siitä, miltä näkymä näyttää heti mitoitus-osuuteen siirryttäessä.

KUVIO 20. Mitoitus osuus

Kuviossa 20 nähdään paljon uusia asioita ja painikkeita. ”Muokkaa piirrosta” -painikkeella päästää takaisin piirto-osuuteen, mikäli profiilin rakenteeseen halutaan tehdä vielä muutoksia. Profiilin nykyinen leikkausleveys, eli yksinkertaisemmin sanottuna profiili yhteenlaskettu pituus voidaan nähdä kuvion 20 oikeassa alakulmassa, ja se on tällä hetkellä 64 mm.

3.2.1 Viivan pituuden muuttaminen

Jokaisen viivan pituus nähdään erikseen aina jokaisen viivan keskellä olevassa oranssireunuksisessa laatikossa. Viivan pituutta on mahdollista muuttaa kahdella eri tapaa. Klikkaamalla oranssireunuksista laatikkoa päästään mitoittamaan kyseistä viivaa. Kuviossa 21 nähdään esimerkki siitä, mitä tapahtuu, kun laatikkoa painetaan.

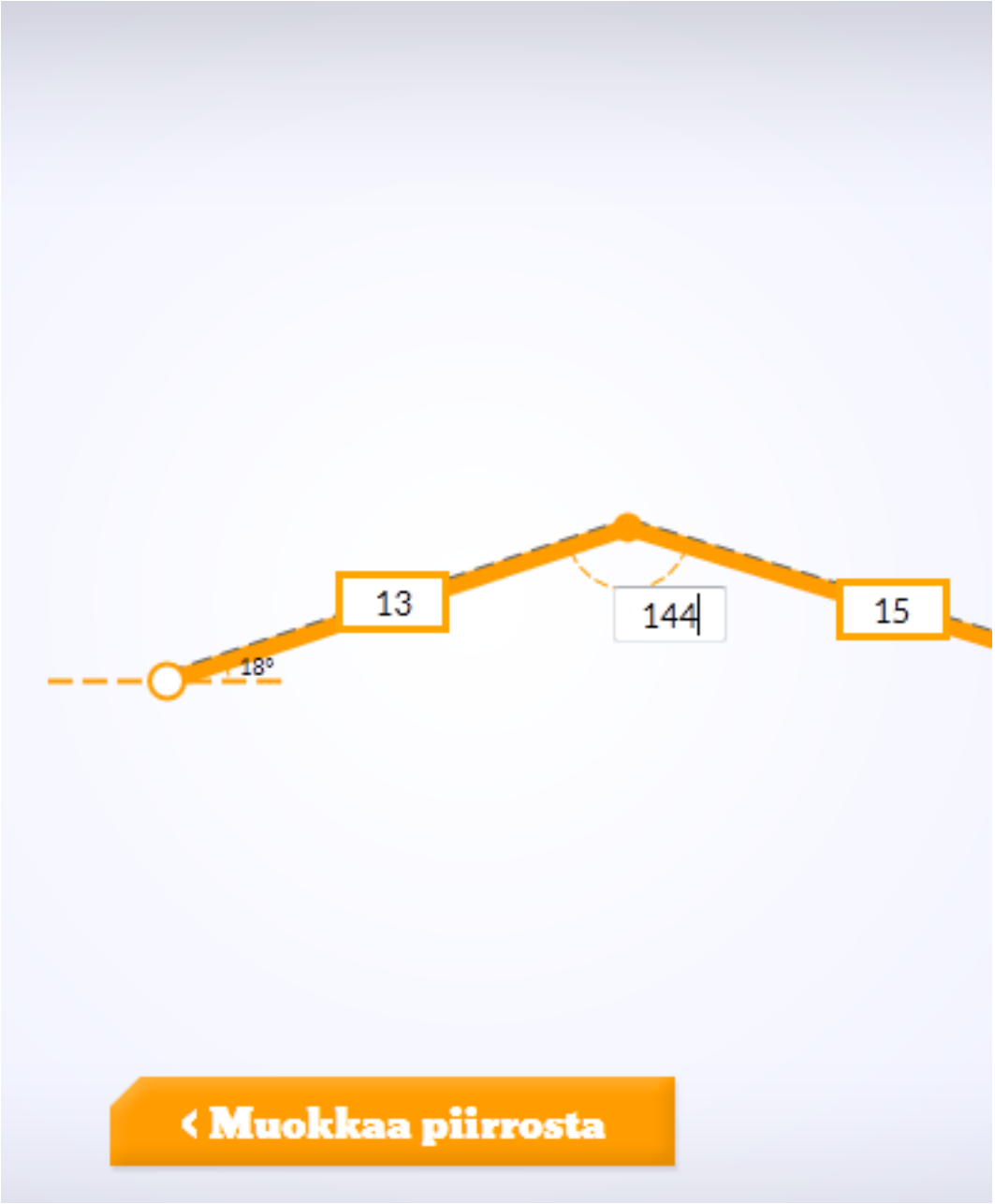
KUVIO 21. Viivan mitoitus

Laatikkoa painaessa esiin tulevat apuviivat ja tässä vaiheessa käyttäjän on mahdollista muuttaa viivan pituus, joko apuviivojen arvoja muuttamalla tai suoraan ison laatikon arvoa muuttaen. Jos käyttäjä muuttaa arvoa isosta laatikosta käsin, niin sovellus laskee uuden päätepisteen vektorille ja järjestää loput vektorit myös tämän loppupisteen mukaan, jottei kuvion muoto muuten kärsi. Tämän jälkeen TweenLite animoi viivan pituuden muuttamisen ja pituus on muutettu. Jos taas käyttäjä päättää muuttaa apuviivojen arvoja, niin pääviivan pituus muuttuu

toki siinä sivussa, mutta myös kantakulma muuttuu mukana. On siis oltava tarkkana ja tiedettävä, mitä tekee, jos lähtee käyttämään apuviivoja. Viivojen pituuksia muuttettaessa sovellus tarkastaa, onko annettu arvo tapeeksi pitkä, ja jos ei, niin numero muuttuu punaiseksi osoittaakseen, että arvo on virheellinen, eikä muutosta tapahdu ennen kuin arvo korjataan.

3.2.2 Kulmien suuruuden muuttaminen

Mitoitus-osuudessa jokaisessa taitekohdassa nähdään myös kulmien suuruudet asteina. Kulmissa on katkoviivat osoittamassa sitä, kumpaa puolta sen vieressä oleva arvo osoittaa. Sovellus ilmoittaa aina pienemmän kulman arvon eli yli 180 asteen kulmia ei nähdä. Jokaisessa profiilin taitekohdassa, eli kulmassa on pallo, jota klikkaamalla kyseisen kulman asteluku muuttuu syöttökentäksi ja kulman suuruuttaa voidaan muuttaa. Kulman syöttökentässä on myös sama ominaisuus kuin viivan pituuden syöttökentässä, eli arvo muuttuu punaiseksi, jos kenttään syötetään virheellinen arvo. Kulmaa muuttaessa koko muu kuvio pysyy ennallaan ja pyörii mukana, eli vain muutettu kulma muuttuu. Profiilin alkukulmaa osoitetaan paperilla olevalla oranssilla katkoviivalla. Kuviossa 22 voidaan nähdä alkukulman viiva ja kulman syöttökenttä.

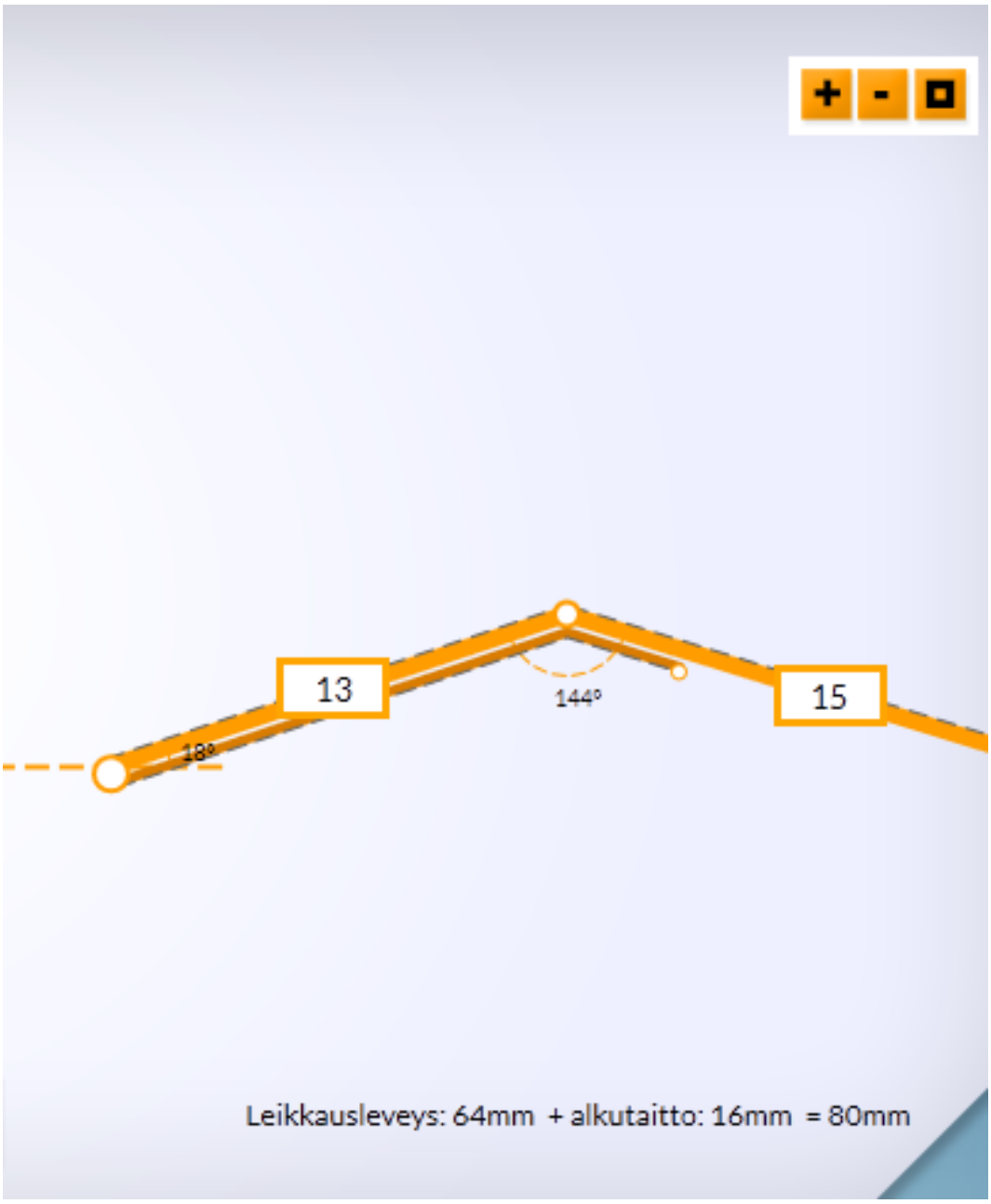


KUVIO 22. Kulman muuttaminen ja alkukulma

3.2.3 Päädytys

Kuviossa 22 nähdään myös alkukulma, jonka piste on huomattavasti normaalia taitekohdan pistettä suurempi. Tämä on siksi, että molempiin profiilin alku- sekä loppupäihin voidaan tehdä niin sanotut päädyt, eli taittaa profiili kulkemaan haluttu matka taaksepäin, jotta saadaan peltien päädyt pyöreiksi. Tällaista päätypalloa klikattaessa pallon viereen tulee painike nimeltä ”päädyt”, jota

klikkaamalla päästään päädyttämään. Päädytyksen ollessa käynnissä sovellus häivyttää viivojen pituuksia näyttävät laatikot pois päädytyksen tieltä ja päädytys voi alkaa. Vetämällä hiirtä profiilin viivaa pitkin aloittaen valitusta päädystä, sovellus piirtää tummemman viivan profiilin alapuolelle, mikä ilmoittaa päädyn pituutta. Päädyt voivat olla minkä mittaisia tahansa, ja ne voivat mennä jopa yli profiilien taitteista. Päädyn ollessa tarpeeksi pitkällä klikataan haluttua paikkaa, minkä jälkeen häivytetyt viivan pituudet tulevat esiin ja päädytys on valmis. Kuviossa 23 nähdään, miltä valmiiksi päädytetty pääty mitoitus-osuudessa näyttää.



KUVIO 23. Päädytys

Kuviossa 23 nähdään myös leikkausleveyteen lisätyn päädytyksen pituus, eli alkutaiton pituus, sekä koko profiilin yhteispituus. Päädyt on mahdollista poistaa sekä päädyttää uudelleen painamalla samaista palloa profiilin alusta tai lopusta. Tässä vaiheessa on myös muistettava, että jos profiilia on päädytetty ja halutaan lähteä uudelleen piirto-osuuteen muokkaamaan omaa profiilia, niin päädyt teknisistä syistä häviävät ja ne joudutaan päädyttämään uudelleen, kun palataan takaisin mitoitukseen.

3.2.4 Pintapuolen värin vaihto

Kuviossa 23 voidaan nähdä sekä profiilin että päädyn pintapuolella harmaata katkoviivaa. Tämä ilmoittaa käyttäjälle peltiprofiilin pintapuolen, eli sen puolen, joka lopputuotteessa tullaan maalaamaan. Pintavärin puoli on mahdollista vaihtaa ”Vaida pintavärin puoli” -painikkeella. Pintavärin puolta vaihdettaessa myös päädyt vaihtavat puolta. Kuviosta 24 voidaan nähdä ero pintavärin vaihdon tapahtuessa ja sitä voidaan verrata kuvioon 23.

KUVIO 24. Pintavärin puolen vaihto

Kuviosta 24 voidaan myös huomata, että lähennä-, loitonna- ja keskitä-painikkeet ovat käytettävissä myös mitoitus-osuudessa ja toimivat samalla tavalla kuin piirto-osuudessa. Toisin kuin piirto-osuudessa, mitoitus-osuudessa koko profiilia on mahdollista liikuttaa mihin suuntaan haluaa painamalla hiiren napin pohjaan jossain kohtaa taustaa ja raahaamalla hiirtä haluttuun suuntaan. Valmis profiili voidaan tallentaa JavaScript-objektiksi, jonka voi myöhemmin palauttaa takaisin sovellukseen uudelleen käyttöä varten. Tässä vaiheessa mitoitus-osuus on melko loppussa. Esimerkki valmiiksi mitoitetusta profiilista voidaan nähdä Kuviossa 25.

KUVIO 25. Lopputulos

Tämän jälkeen sovelluksen tuottamaa JavaScript-objektia ja sen arvoja hyväksikäyttäen on mahdollista syöttää tarvittavat tiedot peltiprofiilien taittoon tarkoitetulle laitteelle, joka sitten taittaa itse peltiprofiilin.

4 YHTEENVETO

Työn tavoitteena oli toteuttaa graafinen suunnittelutyökalu peltiprofiilien suunnittelemiseen ja mitoittamiseen. Työkalun tarkoitus oli lähinnä siirtyä käsin piirretyistä ja mitoitetuista profiileista helpommin ymmärrettäviin ja yhteneväisiin sovelluksella piirrettyihin suunnitelmiin, jotka helpottavat niin ostajan kuin myyjänkin työtä.

Työkalun ensimmäisen version ollessa valmis voidaan todeta, että suunnitellut ominaisuudet saatiin sisällytettyä sovellukseen hyvin ja tämän jälkeen on hyvä siirtyä testaamaan sovellusta enemmän ja viemään sitä eteenpäin asiakkaalta saadun palautteen perusteella. Testauksessa mahdollisesti ilmenneitä vikoja tietenkin tullaan tulevaisuudessa korjaamaan ja olemassa olevia ominaisuuksia hiotaan.

Graafinen suunnittelutyökalu toteutettiin HTML-, CSS- ja JavaScript-tekniikoita hyväksi käyttäen, ja käytetyt tekniikat toimivat tämäntyylisessä modernissa web-sovelluksessa hyvin. Valittu Paper.js-sovelluskehys on ollut korvaamaton valinta piirrettäessä grafiikkaa canvas-elementille.

Paranneltavaakin työkalusta mahdollisesti löytyy, ja matkan varrella on tullut opittua paljon, mutta kovin montaa asiaa en todennäköisesti tekisi toisin, jos lähtisin tekemään työkalua alusta asti uudelleen. Erinäisten arvojen pyöristysten kanssa oli hieman ongelmia, jotka mahdollisesti ratkaisisin eri tavoin, mutta muuten olen työhön todella tyytyväinen.

LÄHTEET(GREENSOCK.COM, 2013)

Greensock.com. 2013. jQuery vs GSAP: Gage Match [viitattu 17.3.2013]. Saatavissa: http://www.greensock.com/jquery/

Imager.fi. 2013. Yritys [viitattu 18.2.2013]. Saatavissa: http://imager.fi/yritys

JSON.org. 2013. Introducing JSON [viitattu 17.3.2013]. Saatavissa: http://www.json.org/

Paperjs.org. 2013a. About [viitattu 17.3.2013]. Saatavissa: http://paperjs.org/about/

Paperjs.org. 2013b. Keyboard Interaction [viitattu 17.3.2013]. Saatavissa: http://paperjs.org/tutorials/interaction/keyboard-interaction/

Scriptographer.org. 2013a. Document Hierarchy [viitattu 17.3.2013]. Saatavissa: http://scriptographer.org/tutorials/document-items/document-hierarchy/

Scriptographer.org. 2013b. Mouse Tool Events [viitattu 17.3.2013]. Saatavissa: http://scriptographer.org/tutorials/interaction/mouse-tool-events/

Scriptographer.org. 2013c. Vector Geometry [viitattu 17.3.2013]. Saatavissa: http://scriptographer.org/tutorials/geometry/vector-geometry/

Scriptographer.org. 2013d. Working with Path Items [viitattu 17.3.2013]. Saatavissa: http://scriptographer.org/tutorials/paths/working-with-path-items/

Wikibooks. 2013a. CSS3 [viitattu 17.3.2013]. Saatavissa: http://fi.wikibooks.org/wiki/CSS3

Wikibooks. 2013b. HTML DOM [viitattu 5.4.2013] Saatavissa: http://fi.wikibooks.org/wiki/HTML_DOM

Wikipedia. 2013a. Ajax (programming) [viitattu 17.3.2013]. Saatavissa: http://en.wikipedia.org/wiki/Ajax_(programming)

Wikipedia. 2013b. Canvas Element [viitattu 17.3.2013]. Saatavissa: http://en.wikipedia.org/wiki/Canvas_element

Wikipedia. 2013c. Cascading Style Sheets [viitattu 5.4.2013]. Saatavissa: http://fi.wikipedia.org/wiki/Cascading_Style_Sheets

Wikipedia. 2013d. HTML [viitattu 5.4.2013]. Saatavissa: http://fi.wikipedia.org/wiki/HTML

Wikipedia. 2013e. HTML5 [viitattu 17.3.2013]. Saatavissa: http://en.wikipedia.org/wiki/HTML5

Wikipedia. 2013f. JavaScript [viitattu 17.3.2013]. Saatavissa: http://fi.wikipedia.org/wiki/JavaScript

Wikipedia. 2013g. jQuery [viitattu 17.3.2013]. Saatavissa: http://en.wikipedia.org/wiki/Jquery

Wikipedia. 2013h. JSON [viitattu 17.3.2013]. Saatavissa: http://en.wikipedia.org/wiki/Json

W3Schools. 2013a. HTML Canvas Reference [viitattu 5.4.2013]. Saatavissa: http://www.w3schools.com/tags/ref_canvas.asp

W3Schools. 2013b. HTML5 [viitattu 17.3.2013]. Saatavissa: http://www.w3schools.com/html/html5_intro.asp

W3Schools. 2013c. jQuery Events [viitattu 17.3.2013]. Saatavissa: http://www.w3schools.com/jquery/jquery_events.asp

W3Schools. 2013d. jQuery Selectors [viitattu 17.3.2013]. Saatavissa: http://www.w3schools.com/jquery/jquery_ref_selectors.asp